2004

# Minimization Of Power Dissipation In Digital Circuits Using Pipelining And A Study Of Clock Gating Technique

Ranganath Panchangam
*University of Central Florida*

# MINIMIZATION OF POWER DISSIPATION IN DIGITAL CIRCUITS USING PIPELINING AND A STUDY OF CLOCK GATING TECHNIQUE

by

RANGANATH PANCHANGAM
B.S. Nagarjuna University, 2001

A thesis submitted in partial fulfillment of the requirements
for the degree of Master of Science
in the Department of Electrical and Computer Engineering
in the College of Engineering and Computer Science
at the University of Central Florida
Orlando, Florida

Spring term
2004

# ABSTRACT

Power dissipation is one of the major design issues of digital circuits. The power dissipated by a circuit affects its speed and performance. Multiplier is one of the most commonly used circuits in the digital devices. There are various types of multipliers available depending upon the application in which they are used. In the present thesis report, the importance of power dissipation in today's digital technology is discussed and the various types and sources of power dissipation have been elaborated. Different types of multipliers have been designed which vary in their structure and amount of power dissipation. The concept of pipelining is explained and the reduction in the power dissipation of the multipliers after pipelining is experimentally determined. Clock gating is a very important technique used in the design of digital circuits to reduce power dissipation. Various types of clock gating techniques have been presented as a case study. The technology used in the simulation of these circuits is 0.35μm CMOS and the simulator used is SPECTRE S.

## ACKNOWLEDGEMENTS

It has been a great opportunity to be a student of Dr.Yuan and work for my thesis under his guidance. Through out the time of my research, Dr.Yuan gave me many valuable suggestions and advices to enhance the quality of my thesis work. I also thank my committee members Dr.Mainak Chatterjee and Dr.Morgan Wang for their support and taking the time to serve in my committee. Also I would like to thank my parents and family for their support and encouragement. I would also like to take this opportunity to thank my friend Mr. Sridhar who gave me the motivation and courage to move on when ever I fell short of it.

Last but not the least I would like to thank the students of CDRL (Chip Design and Reliability Laboratory) for their timely guidance and help.

# TABLE OF CONTENTS

# LIST OF FIGURES

# LIST OF TABLES

# CHAPTER 1
# INTRODUCTION

## 1.1 Motivation

Minimizing power consumption for digital systems involves optimization at all levels of the design. This optimization includes the technology used to implement the digital circuits, the circuit style and topology, the architecture for implementing the circuits and at the highest level the algorithms that are being implemented. The most important technology consideration is the threshold voltage and its control, which allows the reduction of supply voltage without significant impact on logic speed. Since energy is consumed only when capacitance is being switched, power can be reduced by minimizing this capacitance through operation reduction, choice of number representation, exploitation of signal correlations, resynchronization to minimize glitching, logic design, circuit design, and physical design.

Digital multipliers are the most commonly used components in any digital circuit design. They are fast, reliable and efficient components that are utilized to implement any operation. Depending upon the arrangement of the components, there are different types of multipliers available. A particular multiplier architecture is chosen based on the application. The power dissipation in a multiplier is a very important issue as it reflects the total power dissipated by the circuit and hence affects the performance of the device.

The concept of pipelining has been introduced to increase the throughput of the multiplier and hence reduce the power dissipation in the device and ultimately improve the performance of the device. Pipelining reduces the effective critical path by introducing pipelining latches along the data path and thus reduces the average time per instruction. However, pipelining does not reduce the execution time of each instruction, and may increase it.

Clock gating is one of the most important techniques used for the lowering of power dissipation in digital circuits. The main idea is to stop the clock to those portions of the circuit that are idle at a given time. This reduces the switching activity in the registers associated and also the transfer of data from those registers. This leads to a considerable power saving.

## 1.2 Thesis overview

This thesis report details the issues of power dissipation in digital circuits (multipliers) and its minimization using pipelining technique. Chapter 2 gives a detailed description of the various types of power dissipation in CMOS circuits and the various techniques to minimize them. Chapter 3 discusses the various types of multipliers and their advantages and disadvantages. Chapter 4 deals with the concept of pipelining in multipliers and its application in these architectures. Chapter 5 explains the concept of clock gating and the various types of clock gating techniques proposed so far. Chapter 6 gives the experimental results and conclusions arising from the analysis of various multiplier architectures.

# CHAPTER 2
# POWER DISSIPATION

In recent years, the desirability of portable operation of all types of electronic systems has become prominent and a major factor in the weight and size of portable devices is the amount of batteries which is directly impacted by the power dissipated by the electronic circuits. Also, the cost of providing power has resulted in significant interest in power reduction even in non-portable applications which have access to power source. Lately, the significance of power reduction in digital circuits design is being realized over the factors like area minimization and speed.

**Power dissipation** in CMOS circuits arises from two different mechanisms:

**Static power**: This power results from resistive paths from the power supply to the ground. It is independent of frequency and exists whenever the chip is powered on.

**Dynamic power**: This power results from switching capacitive loads between two different voltage states. It is frequency dependent, since no power is dissipated if the node values don’t change.

Even when there are no explicit circuits using static current, the chip will dissipate some static power. The power is the result of leakage current through nominally off transistors.
With small static power, charging and discharging capacitors generally consume most of the power on a CMOS circuit.

## 2.1 Sources of Power Dissipation

There are 3 major sources of power dissipation in digital CMOS circuits, which are summarized as follows:

$$\boxed{P_{avg} = P_{switching} + P_{short\text{-}circuit} + P_{leakage} = \alpha_{0\text{-}1}.C_L.V^2_{dd}.f_{CLk} + Isc.V_{dd} + I_{leakage}.\ V_{dd}}$$

The first term represents the *switching component* of power, where $C_L$ is the load capacitance; $f_{CLk}$ is the clock frequency and $\alpha_{0\text{-}1}$is the node transition activity factor (the average number of times the node makes a power consuming transition in one clock period).

The second term is due to the direct-path *short circuit* current, $I_{sc}$, which arises when both the NMOS and PMOS transistors are simultaneously active, conducting current directly from supply to ground.

Finally, *leakage* current $I_{leakage}$, which can arise from substrate injection and sub-threshold effects, is primarily determined by fabrication technology considerations.

The dominant term in a "well-designed" circuit is the switching component and low power design thus becomes the task of minimizing $\alpha_{0\text{-}1}$, $C_L$, $V_d{}^2{}_d$, $f_{CLk}$ while retaining the required functionality.

### 2.1.1 Switching Power

The switching or dynamic component of power consumption arises when the capacitive load, $C_L$, of a CMOS circuit is charged through PMOS transistors to make a voltage transition from 0 to the high voltage level, which is usually the supply $V_{dd}$. For an inverter circuit, the energy drawn from the power supply for this positive going transition is $C_L.V_d{}^2{}_d$ ; half of which is stored in the output capacitor and half is dissipated in the PMOS device. On the $V_{dd}$ to 0 transition at the output, no charge is drawn from the supply, however the energy stored in the capacitor ($1/2C_L.V^2_{dd}$ ) is dissipated in the pull-down NMOS device. If these transitions occur at a clock rate, $f_{CLk}$, the power drawn from the supply is $C_L.V_d{}^2{}_d\ f_{CLk.}$ However, in general, the switching is best described probabilistically. $\alpha_{0\text{-}1}$ is defined as the average number of times in each clock cycle that a node with capacitance, $C_L$, will make a power consuming transition (0 to 1), resulting in an average switching component of power for a CMOS gate to be,

$$P_{switching} = \alpha_{0\text{-}1}.C_L.V^2_{dd}.f_{CLk}$$

The average transition energy or the ***power-delay product*** can be defined as the amount of energy expended in each switching event or transition and is thus particularly useful in comparing the power dissipation of various circuit styles. If it is assumed that only the switching component of the power dissipation is important, then it is given by

$$\text{Energy per transition} = P_{total}\ /f_{clk} = C_{effective}.V^2_{dd}$$

Where $C_{effective}$ is the effective capacitance being switched to perform a computation and is given by $C_{effective} = \alpha_{0\text{-}1}.C_L$

#### 2.1.1.1 Factors influencing switching component of power

##### 2.1.1.1.1 Logic Function

The type of logic function NAND, NOR or a more complex gate, will all yield different static transition probabilities. Let us first consider a static 2-input NOR gate and assume that only one input transition is possible during a clock cycle and also assume that the inputs to the NOR gate have a uniform input distribution of high and low levels. This means that the four possible states for inputs A and B (00, 01, 10, and 11) are equally likely. For a NOR gate, the probability that the output is 0 is ¾ and that it will be 1 is ¼. The 0 to 1 transition probability of a CMOS gate is given by the probability that the output is in the 0(=3/4) state multiplied by the probability the next state is 1(=1/4). For a NOR gate this translates to

$$\alpha_{0\text{-}1} = p(0).p(1) = p(0)(1-p(0)) = ¾(1-3/4) = 3/16.$$

### 2.1.1.1.2 Logic Style

One basic choice of logic style is between static and dynamic CMOS. For a dynamic implementation of the 2 input NOR gate which is implemented as an NMOS tree with PMOS precharge, power is consumed during the precharge operation for the times when the output capacitor was discharged the previous cycle. Since all inputs are equi-probable, there is then a 75% probability that the output node will discharge immediately after the precharge phase, yielding a transition probability of 0.75. The corresponding probability is smaller for a static implementation.  For the dynamic case, the activity depends on the signal probability, while for the static case the transition depends on previous state. If the inputs to a static CMOS gate do not change from the previous sample period, then the gate does not switch. But for dynamic logic, the gates can switch. Other considerations w.r.t. logic style is the amount of capacitance on the switching node and the sensitivity to supply voltage reduction.

### 2.1.1.1.3 Signal Statistics

The signals in a circuit are typically not always random and it is necessary to consider the effect of signal statistics on power. Consider a 2-input NOR gate and let $P_a$ and $P_b$ be the probabilities that the inputs A and B are 1. The probability that the output node is a 1 is given by:

$$P_1 = (1-P_a) * (1-P_b)$$

Understanding the signal statistics and their impact on switching events can be used to significantly impact the power dissipation.

2.1.1.1.4 Circuit Topology

The manner in which logic gates are interconnected can have a strong influence on the overall switching activity. There are two components to switching activity: a static component (which does not take into account the timing behavior and is strictly a function of the topology and the signal statistics) and a dynamic component (which takes into account the timing behavior of the circuit). To illustrate the point, consider two alternate implementations of F= A.B.C.D. First consider the static behavior assuming that all primary inputs( A,B,C,D) are uncorrelated and random i.e. $P_1(a,b,c,d) = 0.5$.For an AND gate ,the probability that the output is in the 1 state is given by $P_1 = P_a.P_b$

Therefore the probability that the output will make a 0 to 1 transition is given by

$$\alpha_{0\text{-}1} = P_0\, P_1 = (1-P_1)\, P1 = (1- Pa.P_b).Pa.P_b$$

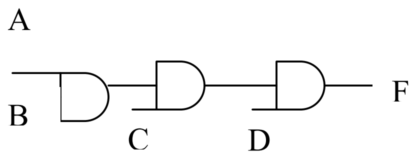


Figure 1: Circuit Topology

Based on this the signal and transition probabilities are computed for the two topologies. The results indicate that the chain implementation will have an overall lower switching activity than the tree implementation for random inputs. However it is also important to consider the timing behavior to accurately make power tradeoffs of various topologies.

Timing skew between the signals can cause spurious transitions (also called *glitches*) resulting in extra power. Based on this issue, when the two above mentioned topologies are compared, the chain implementation causes an extra transition in the output due to timing skew through the logic. On the other hand, the tree implementation is glitch free. This example demonstrates that a circuit topology can have a smaller static component of activity while having a higher dynamic component.

### 2.1.2 Short-Circuit Component of Power

Finite rise and fall times of the input waveforms however result in a direct current path between $V_{dd}$ and GND which exist for a short period of time during switching. Specifically, when the condition $V_{th} < V_{in} < V_{dd} - |V_{tp}|$ holds for the input voltage, where $V_{tn}$, $V_{tp}$ are NMOS and PMOS threshold voltages respectively, there will be a conductive path open between $V_{dd}$ and GND because both the NMOS and PMOS devices will be simultaneously ON. Short circuit currents are significant when the gate input rise and fall times are much larger than the gate output rise and fall times. This is because the short circuit path will be active for a longer period of time. To minimize the total average short-circuit current, it is desirable to have equal input and output edge times. In this case the total power consumed by the short-circuit current is less than 10% of the total dynamic power. An important point to be noted is that if the supply is lowered to be below the sum of the thresholds of the transistors, $V_{dd} < V_{tn} + |V_{tp}|$, the short-circuit currents can be eliminated because both devices will not be ON at the same time for any value of input voltage.

This occurs when the rise/fall time at the input of a gate is larger than the output rise/fall time. To minimize the total average short-circuit current power, it is desirable to have equal input and output edge times. For smaller devices, Short circuit current power is just a linear function of a supply voltage. For larger devices, that are not velocity saturated, the average current is approximately linear with supply voltage so that the average power is a quadratic function of supply voltage. For most ICs, the Short circuit current power is approximately 5-10% of the total dynamic power. If the supply is lowered to below the sum of the thresholds of the transistors, Vdd < $V_{tn}$+ $|V_{tp}|$, however, Short circuit currents will be eliminated because both devices cannot be ON at the same time for all values of input $V_o$.

### 2.1.3 Leakage Component of Power

There are two types of leakage currents:

1) Reverse-bias diode leakage on the transistor drains and
2) Sub-threshold leakage through the channel of an off-device.

The *Diode leakage* occurs when a transistor is turned off and another active transistor charges up/down the drain with respect to the former's bulk potential. For example, consider an inverter with a high input voltage, in which the NMOS transistor is turned ON and the output voltage is driven low. The PMOS transistor will be turned OFF, but its drain –to- bulk voltage will be equal to $–V_{dd}$, since the output voltage is at 0 V and the bulk for the PMOS is at $V_{dd}$.

The resulting current will be approximately $I_L = A_D * J_S$, where $A_D$ is the area of the drain diffusion, and $J_S$ is the leakage current density and weakly dependent on the supply voltage. $J_S$ doubles with every $9^o$ increase of temperature. For a 1 million-transistor chip, assuming an average drain area of 10 μm$^2$, the total leakage current is on the order of 25 μA. While this is typically a small fraction of the total power consumption in MOS chips, it could be significant for a system application, which spends much of its time in standby operation, since this power always being dissipated even when no switching is occurring.

The second component of the leakage power is the *sub-threshold leakage*, which occurs due to carrier diffusion between the source and the drain when the gate-source voltage $V_{gs}$, has exceeded the weak inversion point, but is still below the threshold voltage $V_t$, where carrier drift is dominant. In this regime, the MOSFET behaves similarly as a bipolar transistor, and the sub-threshold current is exponentially dependent on the gate-source voltage $V_{gs}$.

The current in the sub-threshold region is given by

$$I_{ds} = k\, e^{(V_{gs}-V_t)/n.V_T)}\, (1-e^{V_{ds}/V_T})$$

Where k is a function of the technology, $V_T$ is the thermal voltage (kT/q) and $V_t$ is the threshold voltage. For $V_{ds} \gg V_T$, $(1-e^{(V_{ds}/V_T)}) \approx 1$; that is ,the Drain to Source current is independent of the Drain to Source voltage $V_{ds}$, for $V_{ds}$ approximately larger than 0.1v.The *sub-threshold slope* $S_{th}$, in association with this, is the amount of voltage required to drop the sub-threshold current by one decade. The sub-threshold slope can be determined by taking the ratio of two points in this region:

$$I_1/I_2 = e^{(V_1-V_2)/(n.V_T)}$$

Which results in $S_{th} = n*V_T \ln(10)$. At room temperature, typical values of $S_{th}$ lie between 60-90 mV/decade current. Clearly, the lower $S_{th}$ is, the better, since it is desirable to have the device “turn off” as close to $V_t$ as possible as this is one important advantage of silicon–on-insulator technologies which have an $S_{th}$ near 60 mV.

## 2.2 Various techniques for power reduction

### 2.2.1 Technology Optimization

As mentioned earlier, the energy per transition is proportional to $Vdd^2$.Therefore a reduction in the supply voltage can lead to a quadratic improvement in the power-delay product of CMOS logic. But a $V_{dd}$ reduction can lead to a drastic increase in the delay. A simple first order derivation adequately predicts the experimentally determined dependence of delay and $V_{dd}$ and is given by

$$Td = (C_L * V_{dd})/ I = (C_L * V_{dd})/ (u\ Cox)/2\ (W/L)\ (V_{dd} - V_t)^2$$

Since the main objective is to reduce power consumption while maintaining a fixed overall system throughput, a compensation for the increased delays at low voltages is required. One strategy can be using parallel and pipelined architectures to compensate for the increased gate delays at reduced supply voltages. Another approach is to modify the threshold voltage of the devices. Reducing the threshold voltage allows the supply voltage to be scaled down with out loss in speed. For example, a circuit running at a supply voltage of 1.5V with $V_t = 1V$ will have approximately the same performance as the circuit running at a supply voltage of 0.9V and a $V_t = 0.5V$.

But there is limit associated with the reduction of threshold voltage of MOS devices. The limit is set by the requirement to retain adequate noise margins and the increase in sub-threshold currents. The former requirement is relaxed in low power designs, but the latter plays a significant role in power dissipation. The optimal threshold voltage is reached when the sub-threshold leakage current starts increasing with a reduction in supply voltage.

### 2.2.2 Architecture driven voltage scaling

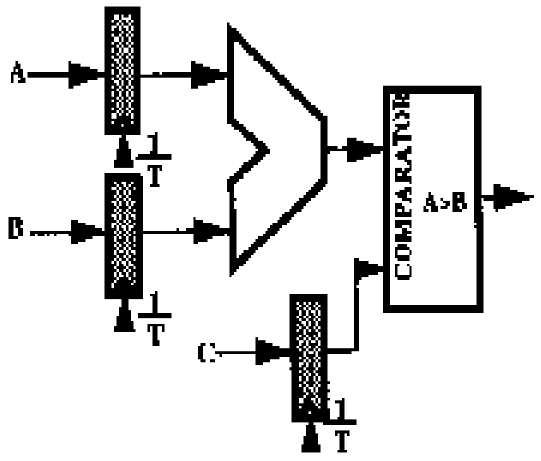


Figure 2: A simple 8-bit data path architecture

An analysis of a simple 8-bit data path consisting of an adder and a comparator is used to illustrate this technique. As per the figure, inputs A and B are added, and the result compared to the worst-case delay through the adder, comparator and latch is approximately 25ns at a supply voltage of 5V. The system can be clocked with a clock period of T = 25ns. The operating voltage cannot be reduced any further at this possible throughput, since no extra delay can be tolerated, hence yielding no reduction in power. Considering this as a reference data path, the power for the reference data path is given by

$$P_{ref} = C_{ref} V^2_{ref} f_{ref}$$

Where $C_{ref}$ is the total effective capacitance being switched per clock cycle. The effective capacitance was determined by averaging the energy over a sequence of input patterns with a uniform distribution. One way to maintain throughput while reducing the supply voltage is to utilize a parallel architecture.

### 2.2.2.1 Parallel Architecture

Two identical adder-comparator data paths are used, allowing each unit to work at half the original rate while maintaining the original throughput. Since the speed requirements for the adder, comparator, and latch have decreased from 25ns to 50ns, the voltage can be dropped from 5V to 2.9 V. while the data path capacitance has increased by a factor of 2, and the operating frequency has correspondingly decreased by a factor of 2. But there is also a slight increase in the total "effective" capacitance introduced due to the extra routing, resulting in an increased capacitance by a factor of 2.15. Thus the power for the parallel data path is given by,

$$P_{par} = C_{par} V^2_{par} f_{par} = (2.15\ C_{ref})\ (0.58\ V^2_{ref})\ (f_{ref}/2) \approx \mathbf{0.36\ P_{ref}}$$

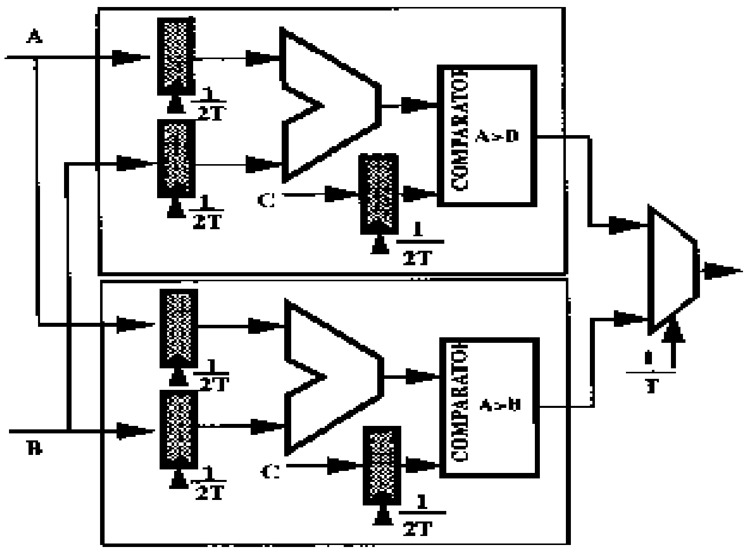


Figure 3: Parallel architecture

The amount of parallelism can be increased to further reduce the power supply voltage and the power consumption for a fixed throughput. However, as the supply approaches the threshold voltage of the devices, the delays increase significantly with a reduction in supply voltage and therefore the amount of parallelism and corresponding overhead circuitry increase significantly. At some "optimum" voltage, the overhead circuitry due to parallelism dominates and the power starts to increase with further reduction in supply.

### 2.2.2.2 Pipelining

From the figure 2.4, with the additional pipeline latch, the critical path becomes the max [$T_{adder}$, $T_{comparator}$], allowing the adder and the comparator to operate at a slower rate. Since the two delays are equal, reducing the supply voltage from 5V to 2.9V will cause no loss in the throughput. However, there is a much lower area overhead incurred by this technique, as we only need to add pipeline registers. There is a slight increase in hardware due to extra latches, increasing the "effective" capacitance by approximately a factor of 1.15.

The power consumed by the pipelined data path is

$$P_{pipe} = C_{pipe} V^2_{pipe} f_{pipe} = (1.15\ C_{ref})(0.58\ V_{ref})^2 f_{ref} = \mathbf{0.66\ P_{ref}}$$

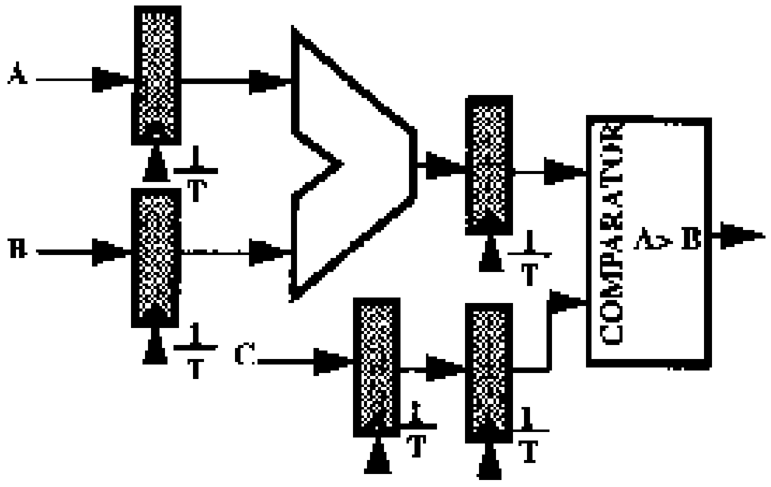


Figure 4: Pipelined architecture

With this architecture, the power reduces by a factor of approximately 2.5, providing approximately the same power reduction as the advantage of lower area overhead. Moreover, increasing the level of pipelining also has the effect of reducing logic depth and hence power contributed due to hazards and critical races.

## 2.3 Process modifications for reducing power

### 2.3.1 Optimizing $V_t$

In most CMOS processes, $V_t$ has been set to a fairly high potential 0.7V to 1.0V. For 5V circuit operation, this has little impact on circuit delay, which is inversely proportional to $(V_{dd} - V_t)^2$. The main benefit for such a large threshold is that the sub threshold leakage is reduced exponentially. While the total leakage current of an IC is still well below the average supply current under operation, the reduced sub threshold current prolongs the duration of stored charge in dynamic circuits, providing more robust operation(due to longer leakage times). Thus, there has been little reason to reduce the thresholds until recently, with the decrease of supply voltages to 3.3 V, and the emphasis on low power design.

Reducing $V_t$ enables the supply voltage to be dropped as well. This maintains circuit speed, but results in a corresponding power decrease. However, the limitation on this is that at low thresholds, the sub-threshold currents become a significant, if not dominant, portion of the average current drawn from the supply.

### 2.3.2 Process Scaling

The primary effect of process scaling is to reduce all the capacitances, which provides a proportional decrease in power and circuit delays. Device sizes can be reduced to keep delay constant over process scaling, which will yield an even larger power dissipation reduction.

### 2.3.3 Fabrication Advancements

With an increased number of metal layers, and a trend towards allowing stacked vias. If these are used judiciously, not only power be reduced, but the circuit area and delay times as well. But circuit redesign is required to utilize these advancements. Using the higher metal layers will help reduce the interconnect capacitance, which contributes roughly one-third to the overall capacitance. By allowing stacked vias, most of the cell areas can be compressed. This also reduces both the intracell and global routing because terminal connections will be closer together; so that most interconnect routes will be reduced in length.

### 2.3.4 Transistor sizing

The transistors of a gate with high fan-out load should be enlarged to minimize the power consumption of the circuit. Transistor sizing is a very effective way to improve the delay of a circuit. With improved process techniques and reduced feature sizes the constraint on area is becoming less strict; so far it has been assumed that power consumption of a circuit is proportional to the "active area" of the circuit. Recent studies have revealed that the power consumption of a static CMOS circuit is not necessarily minimized by minimizing the active area, but, can be improved by enlarging some of the transistors driving large loads.

The power consumption of a single static CMOS gate, neglecting the static power consumption, is composed of $P_{cap}$, the capacitive power due to the charging and discharging of capacitors and $P_{sc}$, the short circuit power dissipated when both the p-type and n-type blocks are conducting simultaneously during a transition. The equivalent width of N series connected transistors for the same delay as that of a single inverter with width W is approximately N x W.

$$W_N = N \times W_1$$

The power consumption of a single static CMOS gate, neglecting the static power consumption, is composed of $P_{cap}$, the capacitive power due to charging and discharging of capacitors and $P_{sc}$, the short circuit power dissipated when both the p-type and the n-type blocks are conducting simultaneously during a transition.

# CHAPTER 3
# MULTIPLICATION PROCESS

Digital multipliers are a major source power dissipation in Digital Signal Processors. High power dissipation in these structures is mainly due to the switching of a large number of gates during multiplication. In addition, much power is also dissipated due to a large number of spurious transitions on internal nodes. Timing analysis of a Full Adder, which is a basic building block in multipliers, has resulted in a different array connection pattern that reduces power dissipation due to the spurious transition activity. Furthermore, this connection pattern also improves the multiplier throughput.

A variety of measures can be used to evaluate the efficiency of the processors. So both the area occupied by the circuit and the time required for the performance of computation must be taken into consideration. Therefore depending on the speed and area requirements, the digital multipliers used can be either of bit-serial or a bit-parallel based architecture. The bit-serial approach processes the data serially where at every clock cycle a single data bit is fed to the processor to be processed. In contrast, the parallel approach processes the data bits in a parallel fashion in just one clock cycle.

The initial architectures were implemented using bit-serial based structures due to their design simplicity and low hardware requirements, which lead to cheap system costs. But for systems that require high speeds, bit-parallel approach is used. But this approach requires large silicon area, communication overhead, and pin out. So a trade off must be made between these two approaches depending on the application.

The multiplication process may be viewed to consist of the following two steps:

1) Evaluation of partial products

2) Accumulation of the shifted partial products.

The product of two n-digit numbers can be accommodated in '2n' digits. In the binary system, an AND gate can be used to generate partial product $X_i Y_j$. The evaluation of the partial products consists of the logical ANDing of the multiplicand and the relevant multiplier bit. Each column of partial products must then be added and, if necessary, values passed to the next column.

In general, the multiplier can be divided into three categories:

1) Serial form.
2) Serial/parallel form.
3) Parallel form.

The speed of the multiplier is determined by both architecture and circuit. The speed can be expressed by the number of the cell delays along the critical path on the architecture level of the multiplier. The cell delay, which is normally the delay of the adder, is determined by the design of the circuit of the cell.

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | | | | $X_3$ | $X_2$ | $X_1$ | $X_0$ | Multiplicand |
| | | | | $Y_3$ | $Y_2$ | $Y_1$ | $Y_0$ | Multiplier |
| | | | | $X_3Y_0$ | $X_2Y_0$ | $X_1Y_0$ | $X_0Y_0$ | |
| | | | $X_3Y_1$ | $X_2Y_1$ | $X_1Y_1$ | $X_0Y_1$ | | |
| | | $X_3Y_2$ | $X_2Y_2$ | $X_1Y_2$ | $X_0Y_2$ | | | |
| | $X_3Y_3$ | $X_2Y_3$ | $X_1Y_3$ | $X_0Y_3$ | | | | |
| $P_7$ | $P_6$ | $P_5$ | $P_4$ | $P_3$ | $P_2$ | $P_1$ | $P_0$ | Product |

Figure 5: 4x4 Multiplication process

## 3.1 Multiplier Architectures

There are three general categories of multiplier architectures: Tree multiplier, Booth-recoded multiplier and Array multiplier.

### 3.1.1 Array multiplier

The array multiplier originates from the multiplication parallelogram. As shown in Figure, each stage of the parallel adders should receive some partial product inputs. The carry-out is propagated into the next row. The bold line is the critical path of the multiplier. In a non-pipelined array multiplier, all of the partial products are generated at the same time. It is observed that the critical path consists of two parts: vertical and horizontal. Both have the same delay in terms of full adder delays and gate delays. For an n-bit by n-bit array multiplier, the vertical and the horizontal delays are both the same as the delay of an n-bit full adder. For a 16-bit by 16-bit multiplier, the worst-case delay is 32 $\tau$ where $\tau$ is the worst-case full adder delay.

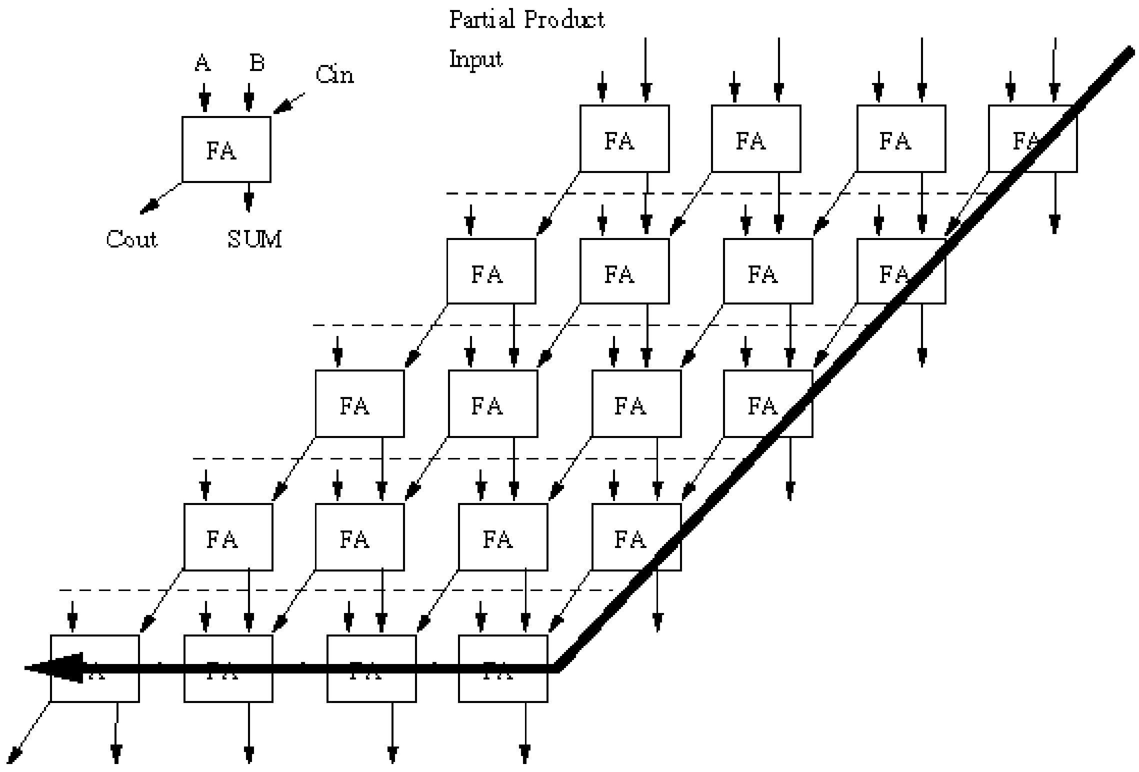


Figure 6: 4x4 Array Multiplier

One advantage of the array multiplier comes from its regular structure. Since it is regular, it is easy to layout and has a small size. The design time of an array multiplier is much less than that of a tree multiplier. A second advantage of the array multiplier is its ease of design for a pipelined architecture. A fully pipelined array of the multiplier with a stage delay equal to the delay of a 1-bit full adder plus a register has been successfully designed for high-speed DSP applications at Bell Laboratories. Also it can be easily pipelined by inserting latches after CSA (carry save adders) or after every few rows.

The main disadvantage of the array multiplier is the worst-case delay of the multiplier proportional to the width of the multiplier. The speed will be slow for a very wide multiplier.

This means that the delay is linearly proportional to the operand size. For example, the worst-case delay of a 54-bit by 54-bit multiplier employing the array scheme will be over 100 $\tau$.

One method to decrease the delay of an array multiplier is to replace the ripple carry adder at the bottom with a carry look-ahead adder. By doing so, the delay can be reduced by roughly 10% for the 8 bit multiplier and 20% for the 32 bit multiplier while increasing the number of gates by about 5% for the 8 bit multiplier and 1% for the 32 bit multiplier.

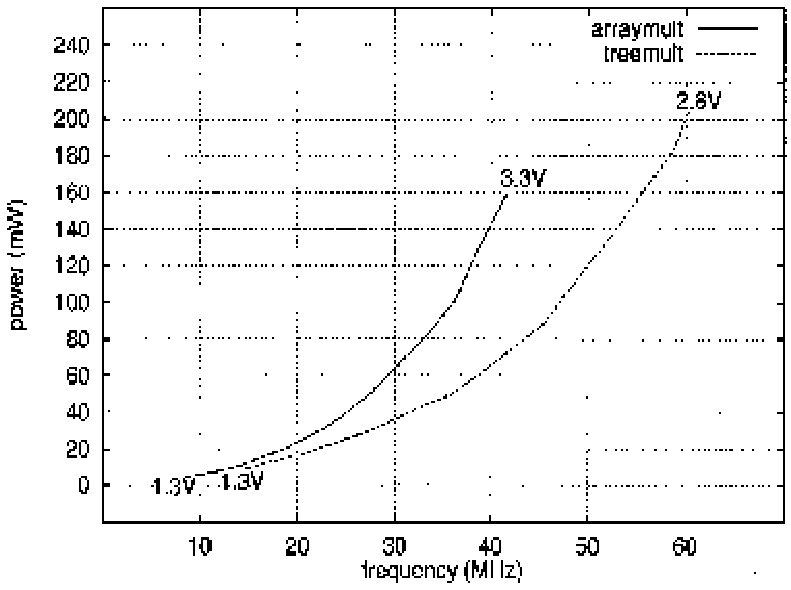


Figure 7: Power versus frequency for the tree and the array multipliers

## 3.1.2 Tree Multiplier

The result of the multiplication is obtained by first generating partial products and then adding the partial products. The critical path of a multiplier depends on the delay of the carry chain through all of the adders. Consider a full adder with A and B as the adder inputs and C is the carry input. The full adder produces a bit of summand and a bit of carry out. It can be observed that a full adder is actually a “one's counter". A, B and C can all be seen as the inputs of 3-2 compressor. The outputs, Carry and Sum, are the encoded output of the three inputs in binary notation .The tree multiplier is based on this property of the full adder. The addition of summands can be accelerated by adopting a 3-2 compressor. A 3-2 compressor adds three bits and produces a two-bit binary number whose value is equal to that of the original three. The advantage of the 3-2 compressor is that it can operate without carry propagation along its digital stages and hence is much faster than the conventional adder.

In any scheme employing 3-2 compressors, the number of adder passes occurring in a multiplication before the product is reduced to the sum of two numbers, will be two less than the number of summands, since each pass through an adder converts three numbers to two, reducing the count of numbers by one. To improve the speed of the multiplication, one must arrange many of these passes to occur simultaneously by providing several 3-2 compressors.

Thus, the best first step for a tree multiplier is to group the summands into threes, and introduce each group into its own 3-2 compressor, thus reducing the count of numbers by a factor of 1.5. The second step is to group the numbers resulting from the first step into threes and again add each group in its own 3-2 compressors. By continuing such steps until only two numbers remain, the addition is completed in a time proportional to the logarithm of the number of summands.

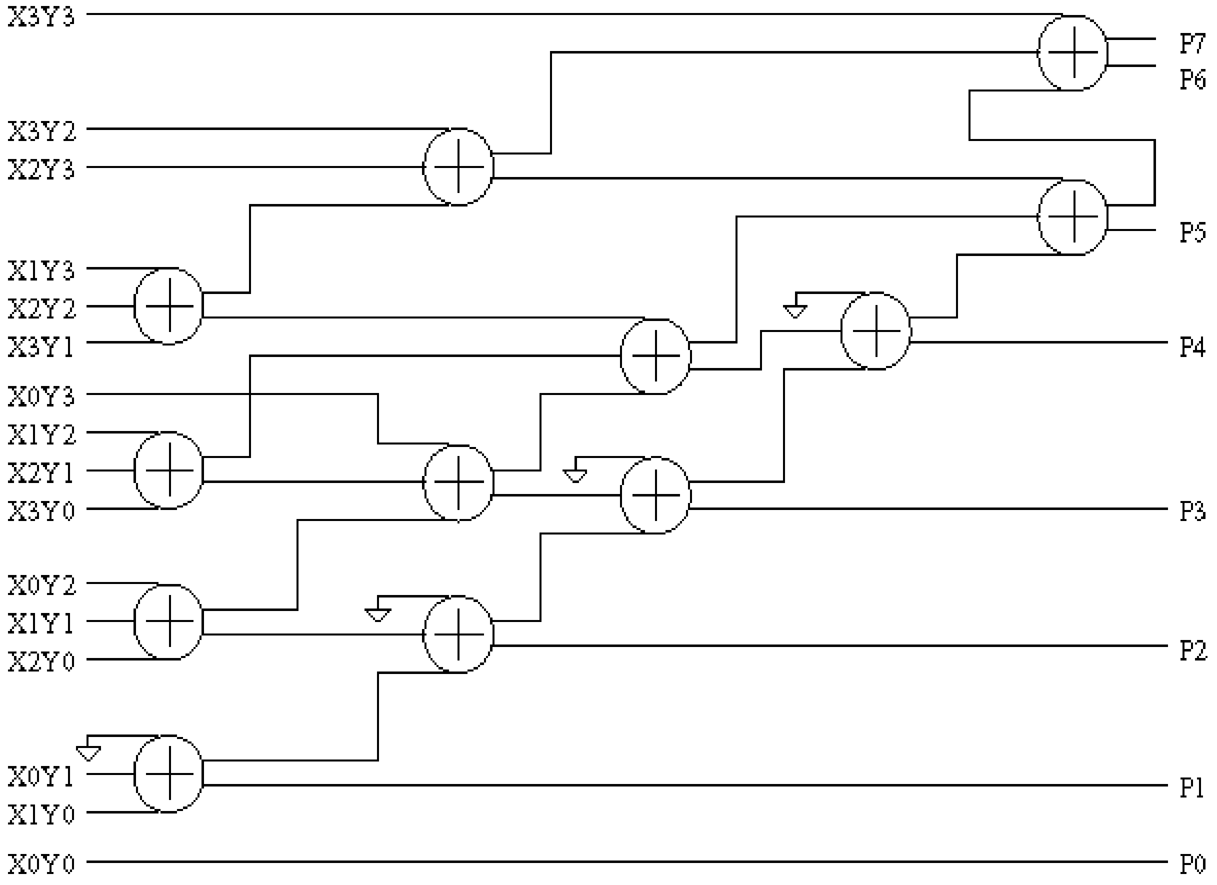


Figure 8: 4x4 Tree multiplier

### 3.1.3 Baugh-Wooley Multiplier

This technique has been developed in order to design regular multipliers, suited for 2's-complement numbers. Let us consider 2 numbers A and B:

$$\mathbf{A} = (\mathbf{a}_{n-1} \ldots \mathbf{a}_0) = -\mathbf{a}_{n-1} . 2^{n-1} + \sum_{0}^{n-2} \mathbf{a}_i . 2^i$$

$$\mathbf{B} = (\mathbf{b}_{n-1} \ldots \mathbf{b}_0) = -\mathbf{b}_{n-1} . 2^{n-1} + \sum_{0}^{n-2} \mathbf{b}_i . 2^i$$

The product A.B is given by the following equation:

$$A.B = a_{n-1}.b_{n-1}.2^{2n-2} + \sum_{0}^{n-2}\sum_{0}^{n-2} a_i.b_i.2^{i+j} - a_{n-1}\sum_{0}^{n-2} b_i.2^{n+i-1} - b_{n-1}\sum_{0}^{n-2} a_i.2^{n+i-1}$$

We see that subtractor cells must be used. In order to use only adder cells, the negative terms may be rewritten as:

$$-a_{n-1}\sum_{0}^{n-2} b_i.2^{i+n-1} = a_{n-1}.\left(-2^{2n-2} + 2^{n-1} + \sum_{0}^{n-2}\overline{b_i}.2^{i+n-1}\right)$$

This way, A.B becomes:

$$A.B = a_{n-1}.b_{n-1}.2^{2n-2} + \sum_{0}^{n-2}\sum_{0}^{n-2} a_i.b_j.2^{i+j}$$

$$+ b_{n-1}\left[-2^{2n-2} + 2^{n-1} + \sum_{0}^{n-2}\overline{a_i}.2^{i+n-1}\right]$$

$$+ a_{n-1}\left[-2^{2n-2} + 2^{n-1} + \sum_{0}^{n-2}\overline{b_i}.2^{i+n-1}\right]$$

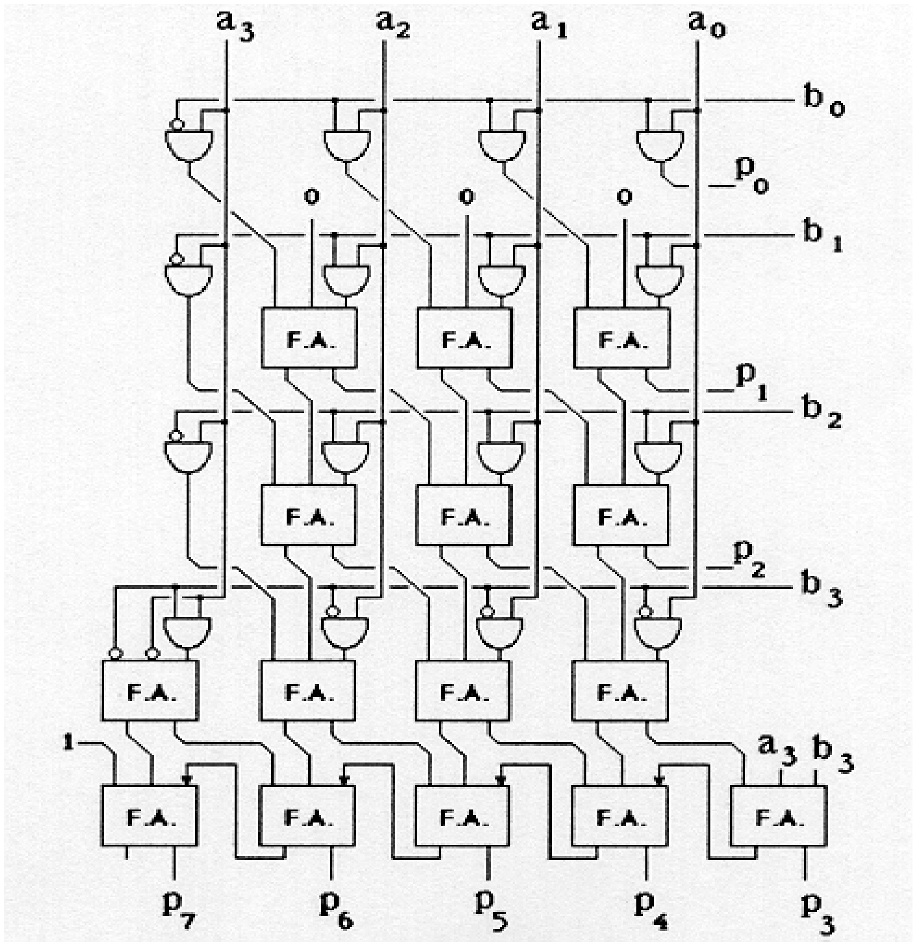


Figure 9: 4x4 Baugh-Wooley multiplier

The final equation is:

$$A.B = -2^{2n-1} + (\overline{a_{n-1}} + \overline{b_{n-1}} + a_{n-1}.b_{n-1}).2^{2n-2}$$

$$+\sum_{0}^{n-2}\sum_{0}^{n-2} a_i.b_j.2^{i+j} + (a_{n-1} + b_{n-1}).2^{n-1}$$

$$+\sum_{0}^{n-2} b_{n-1}.\overline{a_i}.2^{i+n-1} + \sum_{0}^{n-2} a_{n-1}.\overline{b_i}.2^{i+n-1}$$

Because:

$$-(b_{n-1} + a_{n-1}).2^{2n-2} = -2^{2n-1} + (\overline{a_{n-1}} + \overline{b_{n-1}}).2^{2n-2}$$

A and B are n-bits operands, so their product is a 2n-bits number. Consequently, the most significant weight is 2n-1, and the first term $-2^{2n-1}$ is taken into account by adding a 1 in the most significant cell of the multiplier.

### 3.1.4 Wallace tree multiplier

A Wallace tree is an implementation of an adder tree designed for minimum propagation delay. Rather than completely adding the partial products in pairs like the ripple adder tree does, the Wallace tree sums up all the bits of the same weights in a merged tree. Usually full adders are used, so that 3 equally weighted bits are combined to produce two bits: one (the carry) with weight of n+1 and the other (the sum) with weight n. Each layer of the tree therefore reduces the number of vectors by a factor of 3:2. The tree has as many layers as is necessary to reduce the number of vectors to two (a carry and a sum). A conventional adder is used to combine these to obtain the final product. The structure of the tree is shown below. For a multiplier, this tree is pruned because the input partial products are shifted by varying amounts.

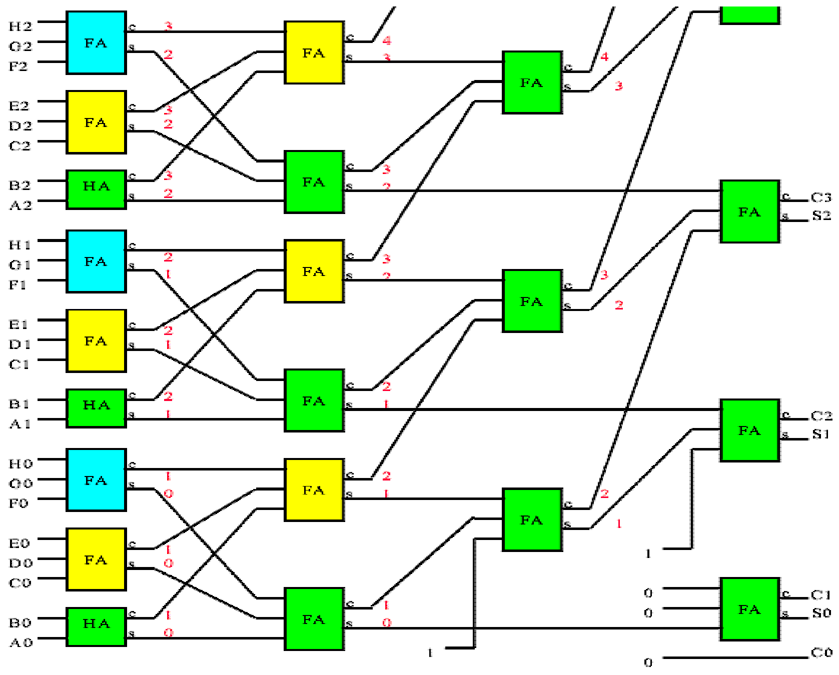


Figure 10: 4x4 Wallace tree multiplier

A section of an 8 input Wallace tree. The Wallace tree combines the 8 partial product inputs to two output vectors corresponding to a sum and a carry. A conventional adder is used to combine these outputs to obtain the complete product.

If the bits in the tree are traced, we will find that the Wallace tree is a tree of carry-save adders arranged as shown below. A carry save adder consists of full adders like the more familiar ripple adders, but the carry output from each bit is brought out to form second result vector rather being than wired to the next most significant bit. The carry vector is 'saved' to be combined with the sum later, hence the carry-save moniker.

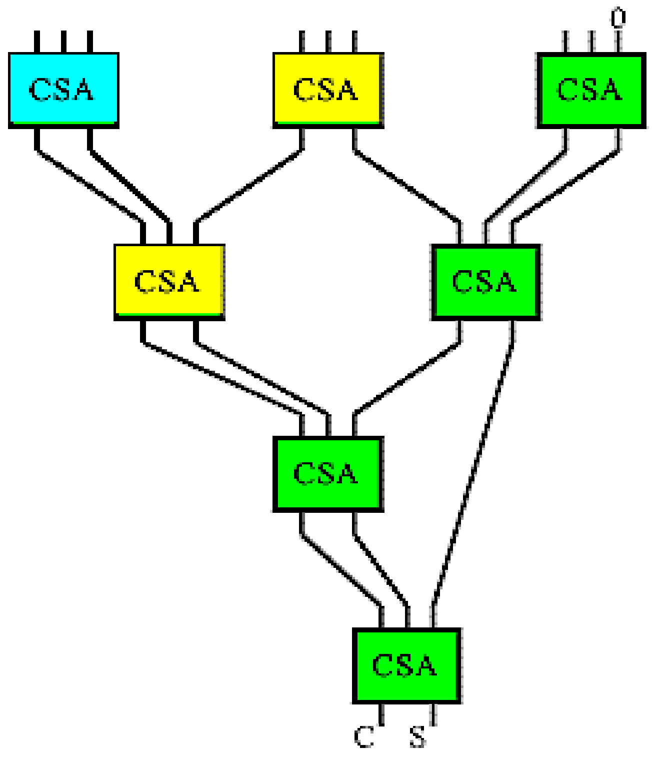


Figure 11: CSA tree structure

A Wallace tree multiplier is one that uses a Wallace tree to combine the partial products from a field of 1x n multipliers (made of AND gates). It turns out that the number of Carry Save Adders in a Wallace tree multiplier is exactly the same as used in the carry save version of the array multiplier. The Wallace tree rearranges the wiring however, so that the partial product bits with the longest delays are wired closer to the root of the tree. This changes the delay characteristic from O (n*n) to O (n*log (n)) at no gate cost.

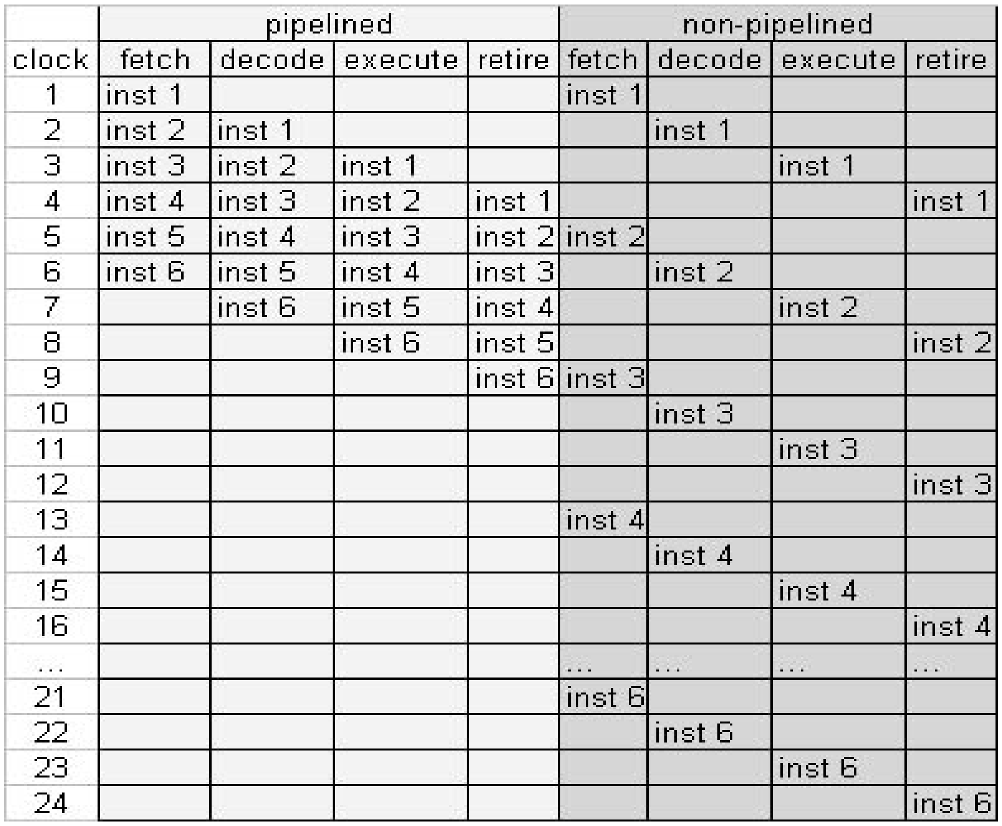
# CHAPTER 4
# PIPELINING CONCEPTS

In computers, a pipeline is the continuous and somewhat overlapped movement of instruction to the processor or in the arithmetic steps taken by the processor to perform an instruction. Pipelining is the use of a pipeline. Without a pipeline, a computer processor gets the first instruction from memory, performs the operation it calls for, and then goes to get the next instruction from memory, and so forth. While fetching (getting) the instruction, the arithmetic part of the processor is idle. It must wait until it gets the next instruction. With pipelining, the computer architecture allows the next instructions to be fetched while the processor is performing arithmetic operations, holding them in a buffer close to the processor until each instruction operation can be performed. The staging of instruction fetching is continuous. The result is an increase in the number of instructions that can be performed during a given time period.

Pipelining is sometimes compared to a manufacturing assembly line in which different parts of a product are being assembled at the same time although ultimately there may be some parts that have to be assembled before others are. Even if there is some sequential dependency, the overall process can take advantage of those operations that can proceed concurrently. Computer processor pipelining is sometimes divided into an instruction pipeline and an arithmetic pipeline. The instruction pipeline represents the stages in which an instruction is moved through the processor, including its being fetched, perhaps buffered, and then executed. The arithmetic pipeline represents the parts of an arithmetic operation that can be broken down and overlapped as they are performed.

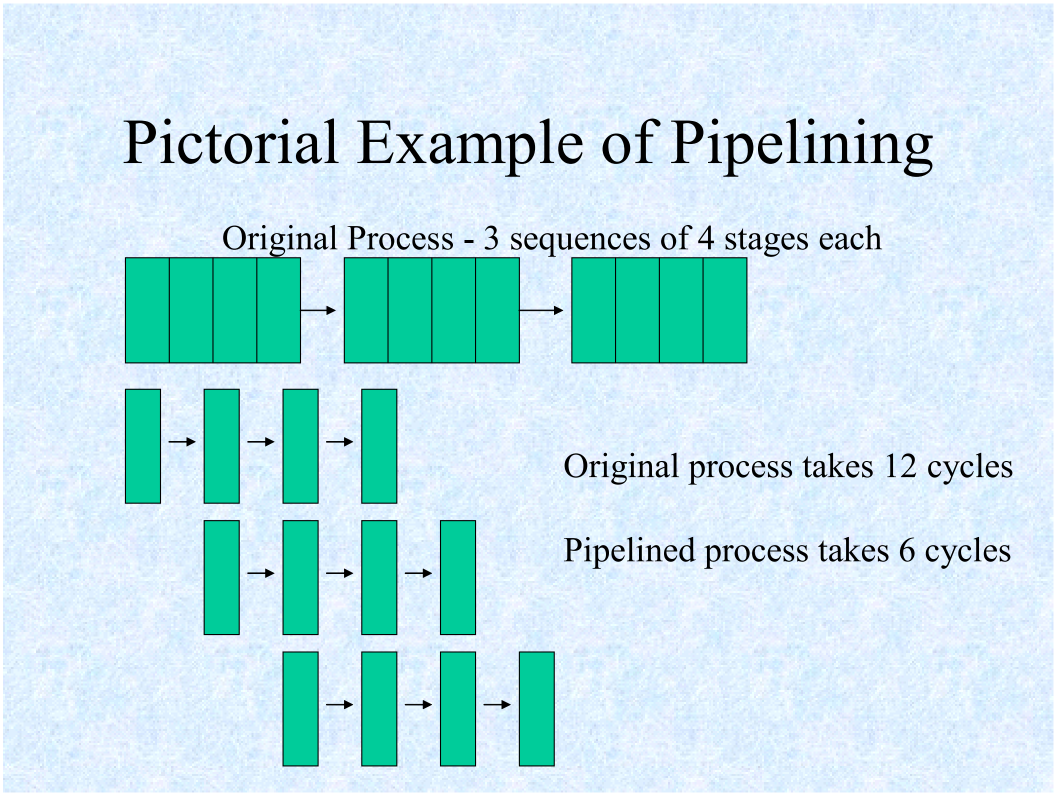
Pipelines and pipelining also apply to computer memory controllers and moving data through various memory staging places. Pipelining reduces the effective critical path by introducing pipelining latches along the data path. Pipelining has been used in the context of architecture design and compiler synthesis. In an M-level pipelined system, the number of delay elements in any path from input to output is (M-1) greater than that in the same path in the original sequential circuit. While pipelining reduces the critical path, it leads to a penalty in terms of an increase in latency. Latency essentially is the difference in the availability of the first output data in the pipelined system and the sequential system. For example, if the latency is one clock cycle, then the k-th output is available in (k+1)-th clock cycle in a 1-stage pipelined system. The two main drawbacks of the pipelining are increase in the number of latches and in system latency.

Table 1: Comparison between pipelined and un-pipelined processor



| | pipelined | | | | non-pipelined | | | |
|---|---|---|---|---|---|---|---|---|
| clock | fetch | decode | execute | retire | fetch | decode | execute | retire |
| 1 | inst 1 | | | | inst 1 | | | |
| 2 | inst 2 | inst 1 | | | | inst 1 | | |
| 3 | inst 3 | inst 2 | inst 1 | | | | inst 1 | |
| 4 | inst 4 | inst 3 | inst 2 | inst 1 | | | | inst 1 |
| 5 | inst 5 | inst 4 | inst 3 | inst 2 | inst 2 | | | |
| 6 | inst 6 | inst 5 | inst 4 | inst 3 | | inst 2 | | |
| 7 | | inst 6 | inst 5 | inst 4 | | | inst 2 | |
| 8 | | | inst 6 | inst 5 | | | | inst 2 |
| 9 | | | | inst 6 | inst 3 | | | |
| 10 | | | | | | inst 3 | | |
| 11 | | | | | | | inst 3 | |
| 12 | | | | | | | | inst 3 |
| 13 | | | | | inst 4 | | | |
| 14 | | | | | | inst 4 | | |
| 15 | | | | | | | inst 4 | |
| 16 | | | | | | | | inst 4 |
| … | | | | | … | … | … | … |
| 21 | | | | | inst 6 | | | |
| 22 | | | | | | inst 6 | | |
| 23 | | | | | | | inst 6 | |
| 24 | | | | | | | | inst 6 |

Pipelining reduce*s the average time per instruction*. Pipelining *does not reduce the* execution time *of each instruction*, and may increase it.



## Pictorial Example of Pipelining

Original Process - 3 sequences of 4 stages each

Original process takes 12 cycles

Pipelined process takes 6 cycles

Figure 12: Pipelining Process

"*Critical path*" in an architecture is defined as the longest path between any 2 latches or between an input and a latch or between a latch and an output or between the input and the output. The critical path can be reduced by suitably placing the pipelining latches in the architecture.

*"Cut-set"* A cut-set is a set of edges of a graph such that if these edges are removed from the graph, the graph becomes disjoint.

Feed-forward cut-set: A cut-set is called a feed-forward cut-set if the data move in the forward direction on all the edges of the cut-set.

The pipelining latches can only be placed across any feed-forward cut set of the graph. The latches on a feed-forward cut set of any FIR filter structure can be placed arbitrarily with out affecting the functionality of the algorithm.

## 4.1 Pipeline Convention

Latency: The delay from when an input is established until the output associated with that input becomes valid.

Throughput: The *rate* at which inputs or outputs are processed is available.

A K-Stage Pipeline ("K-pipeline") is an acyclic circuit having exactly K registers on every path from an input to an output. A combinational circuit is thus a 0-stage pipeline. Every pipeline stage, hence every K-Stage pipeline, has a register on its OUTPUT (not on its input).

The CLOCK common to all registers must have a period sufficient to cover propagation over combinational paths PLUS (input) register $t_{PD}$ PLUS (output) register setup.

The LATENCY of a K-pipeline is K times the period of the clock common to all registers.

The THROUGHPUT of a K-pipeline is the frequency of the clock.

There are two main advantages of using pipelining

- Higher speed
- Lower power

The propagation delay $T_{pd}$ of CMOS circuit is associated with the charging and discharging of the various gate and stray capacitances in the critical path. It can be defined as

$$T_{pd} = C_{charge} V_o / k (V_o - V_t)^2$$

Where $C_{charge}$ denotes the capacitance to be charged/discharged in a single clock cycle, i.e. the capacitance along the critical path, $V_o$ is the supply voltage, $V_t$ is the threshold voltage. Parameter 'k' is a function of technology parameters μ, w/l, and $C_{ox}$.

The power consumption of a CMOS circuit can be estimated using the following equation:

$$P = C_{total} V_o^2 f$$

where $C_{total}$ denotes the total capacitance of the circuit, $V_o$ is the supply voltage, and f is the clock frequency of the circuit.

## 4.2 Pipelining for Low Power

Let the total power consumed by a sequential circuit be represented as

$$P_{seq} = C_{total} V_o^2 f$$

Here $f = 1/T_{seq}$, where $T_{seq}$ is the clock period of the original sequential circuit.

Consider an M-level pipelined system, where the critical path is reduced to 1/M of its original length and the capacitance to be charged/discharged in a single clock cycle is reduced to $C_{charge}/M$. But the total capacitance does not change. If the same clock speed is maintained, i.e. the clock frequency 'f' is maintained, only a fraction of the original capacitance, $C_{charge}/M$, is being charged/discharged in the same amount of time that was previously needed to charge/discharge the capacitance, $C_{charge}$. This implies that the supply voltage can be reduced to $\beta V_o$, where β is a positive constant less than 1. Hence the power consumption of the pipelined filter will be

$$P_{pip} = C_{total}\, \beta^2 V_o^2\, f = \beta^2\, P_{seq}$$

Therefore, the power consumption of the pipelined system has been reduced by a factor of '$\beta^2$' as compared to the original system.

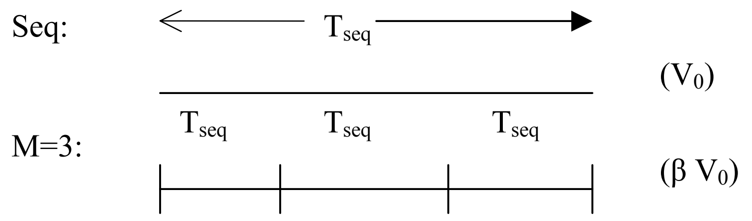


Figure 13: Critical path lengths for original and 3-level pipelined systems

The power consumption reduction factor, β, can be determined by examining the relationship between the propagation delay of the original filter and the pipelined filter.

The propagation delay of the original filter is given by

$$T_{seq} = (C_{charge}\, V_o)/k\, (V_o - V_t)^2$$

The propagation delay of the pipelined filter is given by

$$T_{pip} = (C_{charge}/M)\, \beta\, V_o / k\, (\beta\, V_o - V_t)^2$$

The clock period $T_{clk}$, is usually set equal to the maximum propagation delay, $T_{pd}$, in a circuit. Since the same clock period is maintained for both the circuits, the following quadratic equation can be used to solve for β,

$$M\, (\beta V_0 - V_t)^2 = \beta\, (V_0 - V_t)^2$$

Once β is obtained, the reduced power consumption of the pipelined filter can be computed.

## 4.2 Types of pipelining

The most prominent types of pipelining are wave pipelining and asynchronous pipelining.

### 4.2.1 Wave pipelining

This is a technique employed to decrease the effective number of pipeline stages in a digital system without increasing the number of physical registers in the pipeline. This is achieved by applying new data to the inputs of the combinational logic before the output (due to previous data inputs) is available, thus pipelining the combinational logic. The time of application of new data is determined by performing detailed system-level and circuit-level analysis.

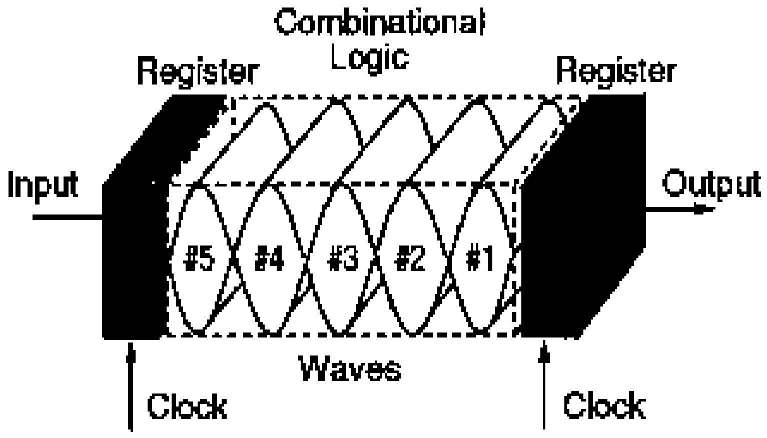


Figure 14: Wave pipelining

At the system level, the analysis should take into account the clock period, the intentional clock skew, and the global clock latency. At the circuit level, precise estimates of the maximum and minimum delays through combinational logic circuits are required. Wave pipelining has a number of advantages. Since no clocked storage elements are inserted in the logic, clocking overhead components such as setup time of registers, clock skew, and the rise and fall time at the output of a logic segment are not added to the latency of the circuit.

Partitioning overhead is eliminated since the circuit need not be segmented. By using fewer storage elements, clock distribution problems are also minimized, reducing the potential clock-skew problems. As a result, a wave pipelined circuit can increase the clock rate while the total latency remains practically unchanged. The main disadvantages are the additional circuitry required for delay equalization and a more substantial performance degradation due to the thermal effects, power-supply level fluctuations, and fabrication process variations.

### 4.2.2 Asynchronous pipelining

This is a technique where there is no global Synchronization within the system; subsystems within the system are only synchronized locally by the communication protocols between them. Systems that involve some sort of request/acknowledge handshake protocols between the subsystems are often referred to as self-timed systems. Since most practical asynchronous systems are self-timed systems, these terms are often used synonymously in the literature. The results produced by subsystems in an asynchronous system can be consumed by other subsystems as soon as they are generated, without having to wait for a global clock change.

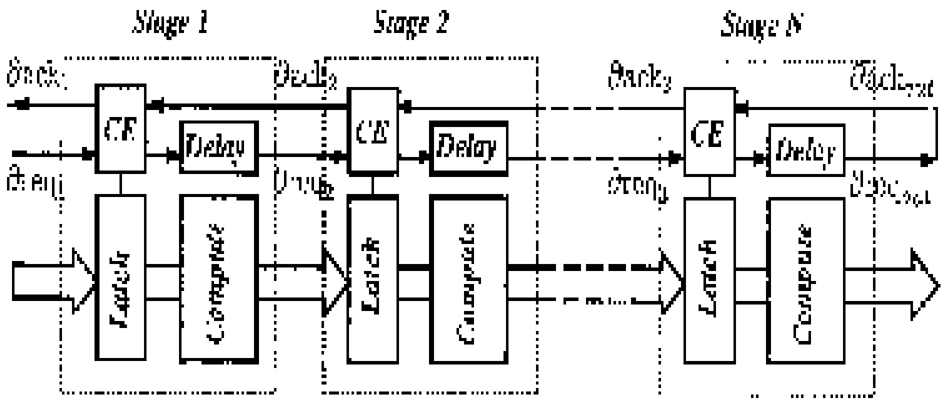


Figure 15: Asynchronous Pipelining

As a result, asynchronous systems are limited by the average-case performance rather than the worst case (as in synchronous systems) of the subsystems. Moreover, a subsystem can be replaced by another subsystem with the same functionality but different performance with ease. However, in synchronous systems this task is not easy as the clock period may have to be recomputed. Finally, transitions in a particular subsystem are only produced when data is processed by it and therefore there is no dynamic power consumption when the subsystem is idle. This makes asynchronous systems attractive for low-power designs. However, the power consumption is increased by the handshaking circuits.

## 4.3 Pipelining Hazards

There are situations, called **hazards** that prevent the next instruction in the instruction stream from being executing during its designated clock cycle. Hazards reduce the performance from the ideal speedup gained by pipelining

There are ***three*** classes of hazards:

### 4.3.1 Structural Hazards

They arise from resource conflicts when the hardware cannot support all possible combinations of instructions in simultaneous overlapped execution. When a machine is pipelined, the overlapped execution of instructions requires pipelining of functional units and duplication of resources to allow all possible combinations of instructions in the pipeline.
If some combination of instructions cannot be accommodated because of a resource conflict, the machine is said to have a structural hazard.

Common instances of structural hazards arise when

- Some functional unit is not fully pipelined. Then a sequence of instructions using that unpipelined unit cannot proceed at the rate of one per clock cycle
- Some resource has not been duplicated enough to allow all combinations of instructions in the pipeline to execute.

Example:

A machine has shared a single-memory pipeline for data and instructions. As a result, when an instruction contains a data-memory reference (load), it will conflict with the instruction reference for a later instruction (instr 3):

Table 2: Structural Hazards

| Clock cycle number | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **Instr** | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| Load | IF | ID | EX | MEM | WB | | | |
| Instr 1 | | IF | ID | EX | MEM | WB | | |
| Instr 2 | | | IF | ID | EX | MEM | WB | |
| Instr 3 | | | | IF | ID | EX | MEM | WB |

To resolve this, the pipeline is stalled for one clock cycle when a data-memory access occurs. The effect of the stall is actually to occupy the resources for that instruction slot. The following table shows how the stalls are actually implemented.

Table 3: Structural Hazards process

| Clock cycle number | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| Instr | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| Load | IF | ID | EX | **MEM** | WB | | | | |
| Instr 1 | | IF | ID | EX | MEM | WB | | | |
| Instr 2 | | | IF | ID | EX | MEM | WB | | |
| Stall | | | | bubble | bubble | bubble | bubble | bubble | |
| Instr 3 | | | | | **IF** | ID | EX | MEM | WB |

Instruction 1 assumed not to be data-memory reference (load or store); otherwise Instruction 3 cannot start execution for the same reason as above.In spite of being a hazard, structural hazard may not be avoided sometimes due to the following reasons:-

- Reduce the cost
- Reduce the latency of the unit

## 4.3.2 Data Hazards

They arise when an instruction depends on the result of a previous instruction in a way that is exposed by the overlapping of instructions in the pipeline.

A major effect of pipelining is to change the relative timing of instructions by overlapping their execution. This introduces data and control hazards. **Data hazards** occur when the pipeline changes the order of read/write accesses to operands so that the order differs from the order seen by sequentially executing instructions on the un-pipelined machine.

Consider the pipelined execution of these instructions:

Table 4: Data Hazards

| | | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|---|
| ADD | **R1**, R2, R3 | IF | ID | EX | MEM | **WB** | | | | |
| SUB | R4, R5, **R1** | | IF | **$\mathbf{ID_{sub}}$** | EX | MEM | WB | | | |
| AND | R6, **R1,** R7 | | | IF | **$\mathbf{ID_{and}}$** | EX | MEM | WB | | |
| OR | R8, **R1**, R9 | | | | IF | **$\mathbf{ID_{or}}$** | EX | MEM | WB | |
| XOR | R10,**R1**,R11 | | | | | IF | **$\mathbf{ID_{xor}}$** | EX | MEM | WB |

All the instructions after the ADD use the result of the ADD instruction (in R1). The ADD instruction writes the value of R1 in the WB stage, and the **SUB** instruction reads the value during ID stage (**$\mathbf{ID_{sub}}$**). This problem is called ***a data hazard***. Unless precautions are taken to prevent it, the SUB instruction will read the wrong value and try to use it.

## 4.3.3 Control Hazards

They arise from the pipelining of branches and other instructions that change the PC. ***Control hazards*** can cause a greater performance loss for DLX pipeline than data hazards. When a branch is executed, it may or may not change the PC (program counter) to something other than its current value plus 4. If a branch changes the PC to its target address, it is a ***taken branch***; if it falls through, it is ***not taken***.

For example, if instruction ***i*** is a ***taken branch***, then the PC is normally not changed until the end of MEM stage, after the completion of the address calculation and comparison.The simplest method of dealing with branches is ***to stall*** the pipeline as soon as the branch is detected until the **MEM** stage is reached, which determines the new PC. The pipeline behavior looks like:

Table 5: Control Hazards

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| Branch | IF | **ID** | EX | **MEM** | WB | | | | | |
| Branch successor | | **IF(**stall) | ***stall*** | ***stall*** | IF | ID | EX | MEM | WB | |
| Branch successor+1 | | | | | | IF | ID | EX | MEM | WB |

The stall does not occur until after **ID** stage (where the instruction is a branch).

## 4.4 Merits and Demerits of pipelining

### 4.4.1 Merits

Pipelining involves introducing the registers in the middle of long combinatorial paths. This adds latency but increases the speed and reduces the levels of logic. The introduction of extra registers consumes power but minimizes the glitches drastically. This glitch reduction has some power advantage. For example, a pipelined 16x16-bit unsigned multiplier generated from ACTGEN Actel's macro-generation utility consumes less power than its unpipelined counterpart.

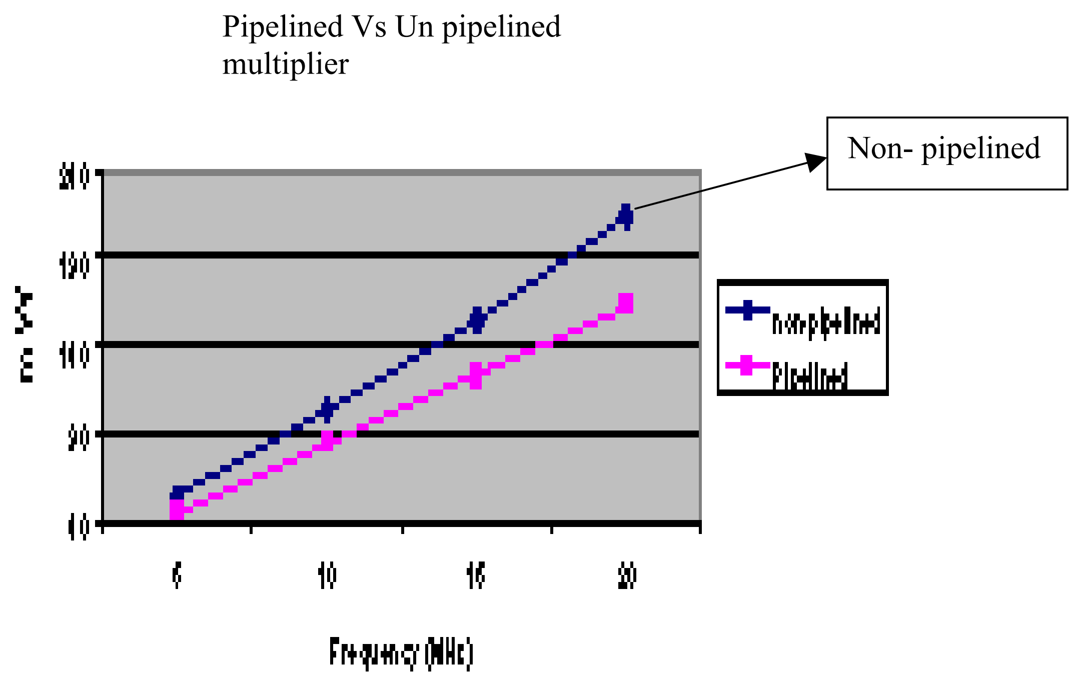


Figure 16: Pipelining Advantage

### 4.4.2 Demerits

One of the more severe penalties for using pipelining concept includes a pipeline stall, by which CPU designers must work around with. If a CPU enters a stall, all the instructions within the pipe must be flushed and the instruction must be re-executed into the CPU. This is problematic for programs with many loops because the CPU will continue to stall until the program is fixed.

Instead of fixing and finding the cause of pipeline stalls, CPU designers instead added more execution units to the ALU of the CPU. Plenty of execution units work in parallel, therefore, leading to an increased throughput (on top of the increased throughput provided by pipelining). It in turn makes a CPU superscalar and powerful since it can process instructions more than one at a time.

Increased execution units, on the other hand, is also problematic because a designer would need plenty of registers to temporarily store executed values. To add hundreds or thousands of registers is not practical since the high-speed architecture that registers utilize is expensive. If designers were to add hundreds or thousands of registers in a CPU, it would become expensive to manufacture. Therefore, caches are used as an alternative.

Adding caches are more practical than using registers because any register value not in use can be received by the caches. Since the caches are physically close to the fetch/decode stage, accessing data from those locations will be quick (although not as quick as using registers).

To further enhance CPU performance, most CPUs utilize a branch prediction table to predict the outcome of a branch. In addition to a pre-fetch mechanism within the CPUs caches, the prediction table and pre-fetch reduce instruction and data transfer latency and therefore, minimize pipeline stalls.

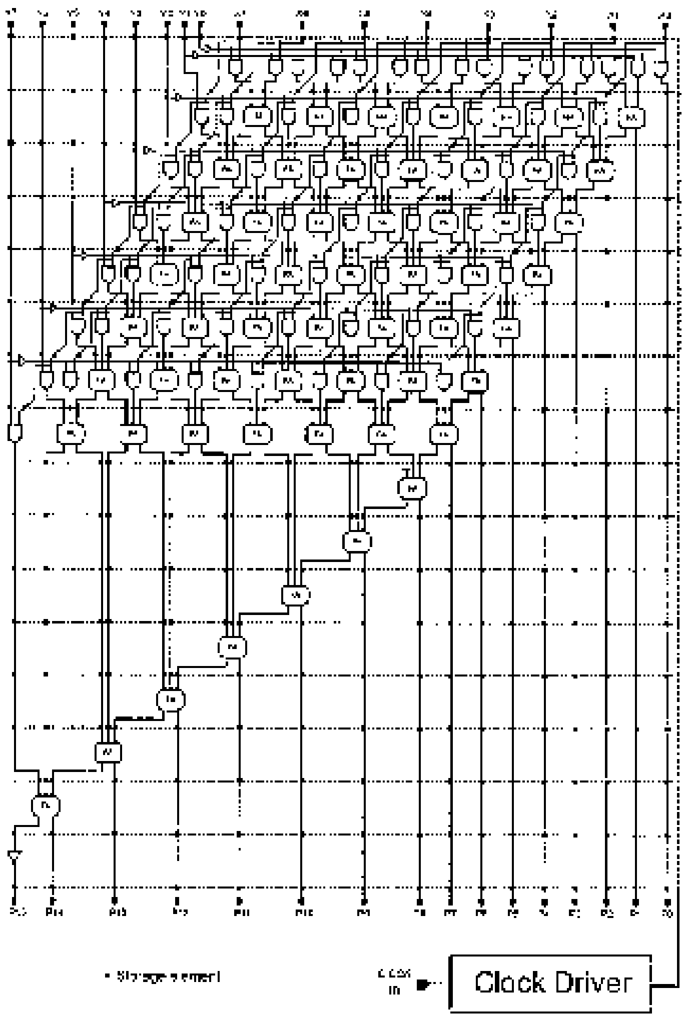


Figure 17: Design of a pipelined multiplier

# CHAPTER 5
# CLOCK GATING

## 5.1 Clock design for low power

The clock could consume 1/3 or more of the total power of a chip. It takes into account the clock generation and distribution as well as the structures of latches and flip-flops .problems such as reliability, clock skew, Meta stability and clock power are more and more crucial in the design of low-power chips in deep submicron technologies. Interconnect delays are well known to be increased in deep submicron technologies, requiring new interconnection strategies, but crosstalk could also be a stopping point. These problems are strongly related to the design of drivers, transistor sizing and finally to layout. This is a difficult problem as reliability in deep submicron technologies is mandatory, but these techniques have also to be capable of reducing the power consumption.

Micro processors are clocked by a global clock signal. The largest difficulty is to have a high frequency global clock and to design the clock tree for such a frequency. This is due to the clock skew that occurs in the clock tree. The proposed solutions for this drawback are to balance the clock tree delays while using various tricks like wire snaking, balance of the number of buffers, of the number of crossovers etc. these solutions generally increase the power consumption. Furthermore, they are not acceptable for very low voltage circuits, as slightly $V_{th}$ variations can change significantly the delay values of the clock buffers. Large hierarchical buffers are used for clock distribution. Clock gating is performed in such clock trees. It reduces the power consumption but introduces some extra clock skew. In most microprocessors, delayed clock are created by analog delays to have more clock edges to perform all the work in a single pipeline stage.  These clocks are delayed to provide time borrowing.

Some of the techniques capable of reducing clock power consumption are as follows:-

- Hierarchical clock tree
- Careful design of the clock drivers to avoid the short circuit current
- Minimum number of MOS connected to the clock (careful design of D-flip-flops and latches)
- Frequency reduction by using double edge flip-flops or parallelized logic modules
- Multi rate clocks, i.e. to clock some blocks at f/2, f/4….

## 5.2 Clock gating

*Clock gating* is a well-known technique to reduce clock power. Because individual circuit usage varies within and across applications, not all the circuits are used all the time, giving rise to power reduction opportunity. By ANDing the clock with a gate-control signal, clock gating essentially disables the clock to a circuit whenever the circuit is not used, avoiding power dissipation due to unnecessary charging and discharging of the unused circuits. Specifically, clock gating targets the clock power consumed in pipeline latches and dynamic-CMOS-logic circuits used for speed and area advantages over static logic. Effective clock gating, however, requires a methodology that determines which circuits are gated, when, and for how long.

Clock-gating schemes that either result in frequent toggling of the clock-gated circuit between enabled and disabled states, or apply clock gating to such small block that the clock-gating control circuitry is almost as large as the block itself, incur large overhead. This overhead may result in power dissipation *higher* than that without clock gating. This is the most widely used practice for power reduction to stop the clock whenever the device is not in use. Clock gating can be applied to sub-sections of the design as well as the whole device.

However, correctly stopping the clock is very important. The gating signal and logic should be generated properly to eliminate any glitch on the clock line. Also, the gating logic adds some delay to the clock, which affects setup and hold times. While using clock gating, the user should take care of the placement of gating logic to minimize delay in the clock network.

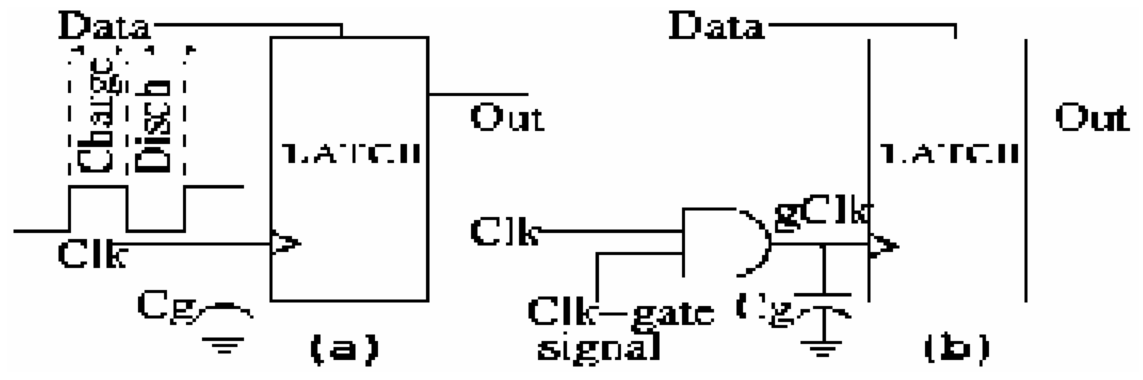


Figure 18: Clock gating a latch element

The above figure shows the schematic of a latch element. $C_g$ is the latch's cumulative gate capacitance connected to the clock. Because the clock switches every cycle, $C_g$ charges and discharges every cycle and consumes significant amount of power. Even if the inputs do not change from one clock to the next, the latch still consumes clock power. In figure (b), the clock is gated by ANDing it with a control signal, which is referred to as *Clk-gate signal*. When the latch is not required to switch state, *Clk-gate signal* is turned off and the clock is not allowed to charge/discharge $C_g$, saving clock power. Because the AND gate's capacitance itself is much smaller than Cg, there is a net power saving.

While gating the clock can provide enormous gains in the power efficiency of a design, the addition of clock gating complicates the clock signal distribution. Gated groups are developed from the logical model of the circuit. An effective low power gated clock tree tool must balance aspects of the logical and physical design in order to reach the best solution. Gated clock distribution in principle consumes low power due to the reduced switching, but depending on the physical placement and routing, the wiring can increase more than the savings from gating the clock. Clock gating is a circuit architecture-level power reduction technique.

Example of clock gating technique using pipelining is shown below

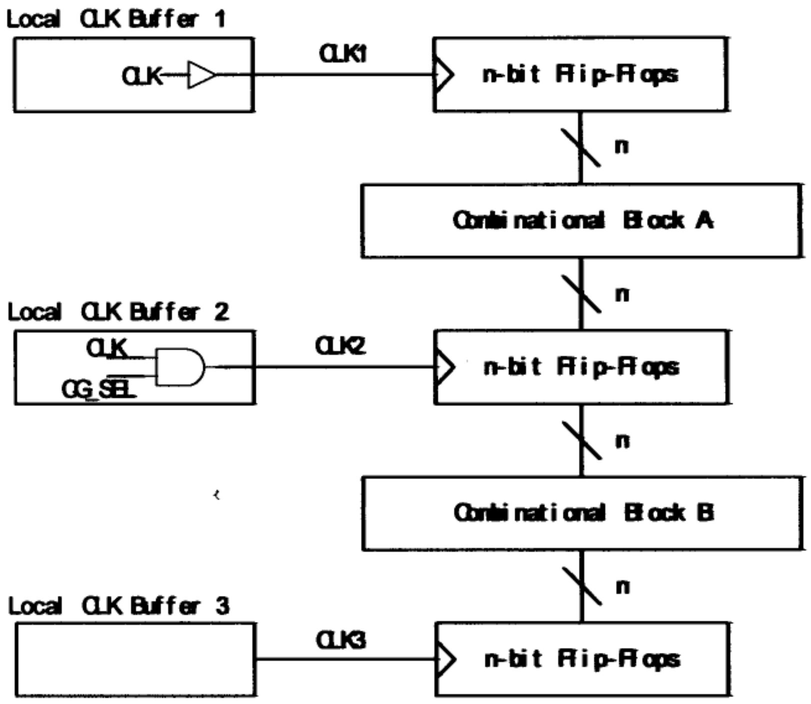


Figure 19: Clock gating example

A local clock buffer 1 is not clock gated and always fires clock signal, CLK1, which feeds n-bit flip-flops. A local clock buffer 2 is clock-gated and if CG_SEL signal is logic low then CLK2 is not fired. Hence power consumption from CLK2 distribution and n-bit flip-flops can be neglected. Furthermore, dynamic power consumption of the block (combinational block B) following the clock-gated flip-flops can be saved during the clock-gating period.

## 5.2 Various clock gating mechanisms

### 5.2.1 Clustered Multiplication

In this technique, an NxN multiplication circuit can be broken into small clusters of N/2 bit multipliers. Each cluster multiplies a nibble from one operand with another nibble from the other operand. If any of these clusters can be disabled, then a significant amount of power will be saved.

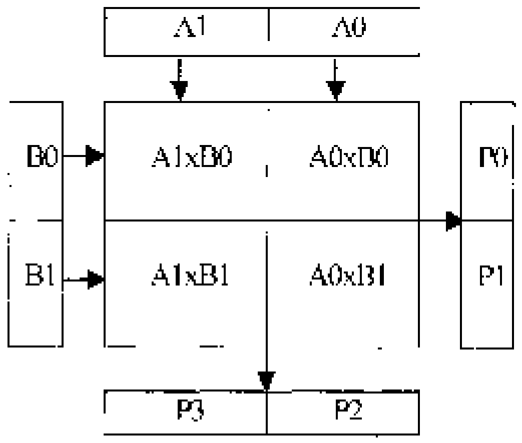


Figure 20: Block diagram of 4 clusters Multiplier

Each multiplication cluster can be implemented using a linear carry save adder array which is power efficient for multiplication of operands of 8-bit or less. As the probability of containing zero nibbles in the input data increases, the amount of saved power increases. A significant power savings of about 13.4% were obtained in [10] using this technique.

### 5.2.2 2-D pipeline gating

In this technique, the clock to the registers is gated in both vertical direction (data flow direction in pipeline) and horizontal direction (within each pipeline stage). The additional area cost to implement this technique is very little and the overhead is hardly noticeable.

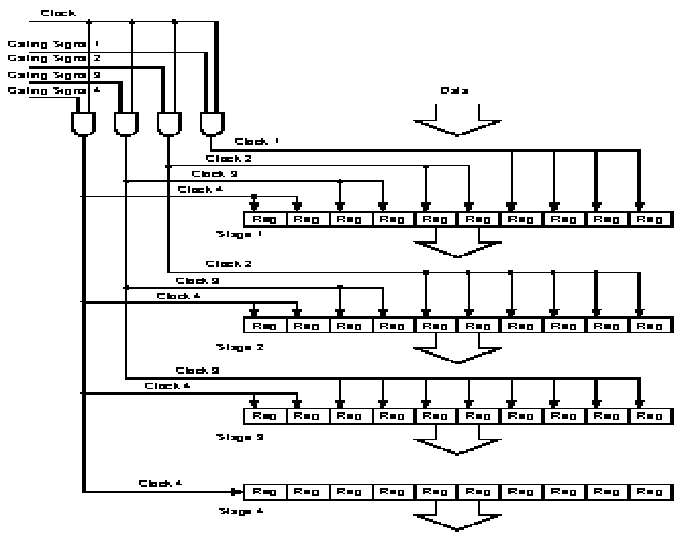


Figure 21: 2-D Pipeline Gating

The effectiveness can be pre-calculated and increases with the growth of the multiplication length. Results from [1] indicate that there is an average power saving of about

66% and an average latency reduction of 47% over the original design under equal input precision probabilities.

### 5.2.3 Deterministic clock gating (DCG)

Clock power is a major component of microprocessor power mainly because the clock is fed to most parts of the circuit and the clock switches every cycle. Usually, the clock power is about 30-35% of the microprocessor power. This technique of deterministic clock gating is based on the observation that for many stages of the pipeline, a circuit block's usage in a specific cycle in the near future is deterministically known a few cycles ahead of time. DCG uses this concept in its working. The main advantages of using this technique are 1.) It clock gates at finer granularities of a few (1-2) cycles and smaller circuit blocks. 2.) It does not use extra heuristics and is significantly simpler. Results from [12] indicate a significant power savings of about 20% implementing this technique.

## 5.3 Latch based clocking scheme

The design methodology using latches and two non-overlapping clocks has many advantages over the use of D-flip-flops methodology with a single phase clock. Due to the non-overlapping of the clocks as long as half period of the master frequency and the additional time barrier caused by having two latches in a loop instead of one D flip-flop , latch based designs support greater clock skew before failing than a similar D flip-flops design. This allows the synthesizer and router to use small clock buffers and to simplify the clock tree generations, which will reduce the power consumption of the clock tree.

With latch-based designs, the clock skew becomes relevant only when its value is close to the non-overlapping of the clocks. When working at lower frequency and thus increasing the non-overlapping of clocks, the clock skew is never a problem. It can even be safely ignored when designing circuits at low frequency. A shift register made with D flip-flops can have clock skew problems at any frequency.

Also, if the chip has clock skew problems at the targeted frequency after integration, the clock frequency of the latch-based design can be reduced. It results that the clock skew problem will disappear, allowing the designer to test the chip functionality and eventually to detect other bugs or to validate the design functionality. This can reduce the number of test integration needed to validate the chip. With a D flip-flop design, when a clock skew problem appears, it needs to be rerouted and integrated again.

Using latches for pipelining is also very good for power consumption when using such a scheme in conjunction with clock gating. The latch design has additional time barriers, which stop the transitions and avoid unnecessary propagation of signal and thus reduce power consumption. The clock gating of each stage of the pipeline with individual enable signals can also reduce the number of transition in the design compared to the equivalent D flip-flops design.

Another advantage with a latch design is the time borrowing. It allows a natural repartition of computation time when using pipeline structures. With D flip-flops, each stage of logic of the pipeline should ideally use the same computation time, which is difficult to achieve, and eventually the design will be limited by the slowest stage. With latches, the slowest pipeline stage can borrow time from either or both the previous and next stage pipeline stage. And the clock skew only reduced the time that can be borrowed.

## 5.4 Clocked systems

The selection of a particular clocking strategy influences how many transistors are used per storage element and how many clock signals need to be routed through out the chip. These decisions impact the size of the chip and the power dissipated by the chip. Hence one of the most important decisions that can be made at the beginning of a design is the selection of a clocking strategy.

Usually latches and registers are used to implement clocked systems. The set-up and hold times of a register are deviations from an ideal register caused by finite circuit delays. The hold time relates to the delay between the clock input to the register and the storage element. i.e. the data has to be held for this period while the clock travels to the point of storage. The set-up time is the delay between the data input of the register and the storage element. As the data takes a finite time to travel to the storage point, the clock cannot be changed until the correct data value appears. In a synchronous system, if the data input to a register does not obey the set-up and hold time constraints, then potential *clock race* problems may occur. These races result in erroneous data being stored in registers. For instance, if the data violates the hold time violation, the data changes to a new value before the data assumes the correct value. With a set-up time constraint the clock changes before the data assumes the correct value.

If the data and the clock do not satisfy the set-up and hold time constraints of a register, a synchronization failure may occur. This is due to the inherent analog nature of the storage elements used in the electronic circuits. A latch with a clock deasserted is normally a bi-stable device; i.e. it has two stable states (0 &1). Under right conditions the latch may enter a meta-stable state. Here the output is in an indeterminate state between 0 and 1. At the CMOS circuit level, this means that the sampled input to an inverter that is responsible for determining the state is close to the inverter threshold voltage. Thus in effect the latch is perfectly balanced between making a decision to resolve a 1 or 0.

With in a synchronously clocked system, as long as the clock to output delays are longer than the set-up times, synchronization failures cannot occur. However, at the boundary of two independently clocked systems or a clocked system and the asynchronous real world, synchronization problems may occur. To deal with these asynchronous interfaces, circuits called *synchronizers* are used at the interface between independently clocked systems. These lower the risk of synchronizer failure to acceptable levels.

## 5.5 Clock skew

In synchronous systems, clock skew is a problem that grows with the speed and size of the system. As a result, in very high-speed bit level pipelined systems this becomes a major bottleneck. Therefore, it is necessary to understand thoroughly the effects of clock skew before designing bit-level pipelined systems. In addition, the way the clock skew affects the operation of a system generally depends on the type of clocking style used and on the direction of data transfer with respect to the clock signal.

Consider a combinational logic block (CLB) which can be a single logic block, a full adder performing bit-level operations, or a multiplier performing word-level operations. A CLB along with a register forms a module or 1 stage of the pipeline.

The propagation delay is the time it takes before at least one of the outputs of the CLB starts a transition in response to a change at any one of its inputs. This delay is typically much shorter than the settling time of the CLB and plays an important role in the clock skew problem.

## 5.6 Clock Distribution

The factors that determine the clock skew in a synchronous digital system are follows:

- The resistance, inductance and capacitance of the interconnection material used in the clock distribution network.
- The shape of the clock distribution network.
- Fabrication process variation over the area of the chip or the wafer.
- Number of processing elements in the digital system and the load presented by each module to the clock distribution network.
- Rise and fall times and the frequency of the clock signal.
- Buffering schemes and clock buffers used.

In order to study these effects, a distributed RC model for the clock distribution network can be used. The most widely studied clock distribution network that minimizes skew is called as the H-tree network. The architecture consists of 8x8 array of cells identical in size. Since the cells are identical, they present the same load and the paths from the clock source to all the cells are similar and as a result, the clock skew is minimized. For an N X N array of processing elements the clock pulse rise time and the clock skew associated with it is O ($N^3$).

Therefore, it appears that as N increases, the clock skew would increase rapidly and become a stumbling block. However, this problem can be overcome by using a distributed buffering scheme. One of the disadvantages of introducing buffers in the H-tree network is the area overhead. The other disadvantage is the increase in the sensitivity of the design to process variations. This is however offset by the improvement gained in speed. H-tree networks are most useful when all processing elements in the system are identical and the clock can be truly distributed in a binary fashion. An alternative to the H-tree network suitable for on-chip clock distribution is the network using a 2-level buffering scheme. This technique has been used to design bit-level pipelined multiply-accumulate systems.

## 5.7 Inductive noise in circuits due to clock gating

Ground bounce which is also known as simultaneous switching noise or delta-I noise, is a voltage glitch induced at power/ground distribution connections due to switching currents passing through the wire/substance inductance (L) associated with power or ground rails. These voltage glitches, which are proportional to L $\delta I/\delta t$, result in large currents that charge/discharge the power/ground buses as the frequency and number of gates increase, which leads to ground bounce in a short time. Also with a drop in supply voltages, the noise margin of circuits also decreases. If the magnitude of the voltage surge/droop due to ground bounce is greater than the noise margin of a circuit, the circuit may erroneously latch the wrong value or switch at the wrong time.

Clock gating methods to reduce power consumption of circuitry, essentially involve ANDing the clock feeding a unit with a control signal, which shuts down the unit when not in use. However, rapid power down implies rapid switching and hence enhanced ground bounce. Moreover, the powered down unit may not become ready on time, which could cause synchronization problems. In order to avoid the problem, the basic idea implemented is to increase the time for the units to switch on/off. This causes them to draw a lesser instantaneous current and hence reduces the glitch in the power/ground lines. However, since it now takes a finite amount of time for a resource to become available, additional stalls might occur in the execution pipeline. This will in turn effect the instructions/cycle (IPC) executed. But this loss is considered small for modern day processors.

A clock-gating scheme is proposed in [], that uses a tool called simple scalar tool, to measure the performance change of a processor due to this method of clock gating. Studies indicate that there is a 2x reduction in the $\delta I/\delta t$ noise at the architectural level at a modest cost in hardware and performance.

# CHAPTER 6
# CONCLUSIONS

In the present thesis work, various multiplier circuits have been designed and implemented to study the variation of power dissipation in these circuits. Multipliers are a major source of power dissipation in digital circuits. The concept of pipelining has been implemented in the design and the reduction in the power dissipation in the same circuits has been noticed. The technology used is 0.35 µm CMOS process, using CADENCE tool and SPECTRE S simulator. On an average, there is a power savings of about 40%. Also the concept of clock gating has been thoroughly studied and presented even though it was not implemented in these circuits. Based on the experiments and plotting of the graphs, it has been concluded that power dissipation plays a major role in the performance of modern day digital circuits. And it has been predicted that as the technology improves, and the design of circuits is modified, a significant savings in the power dissipation of the circuits is possible.

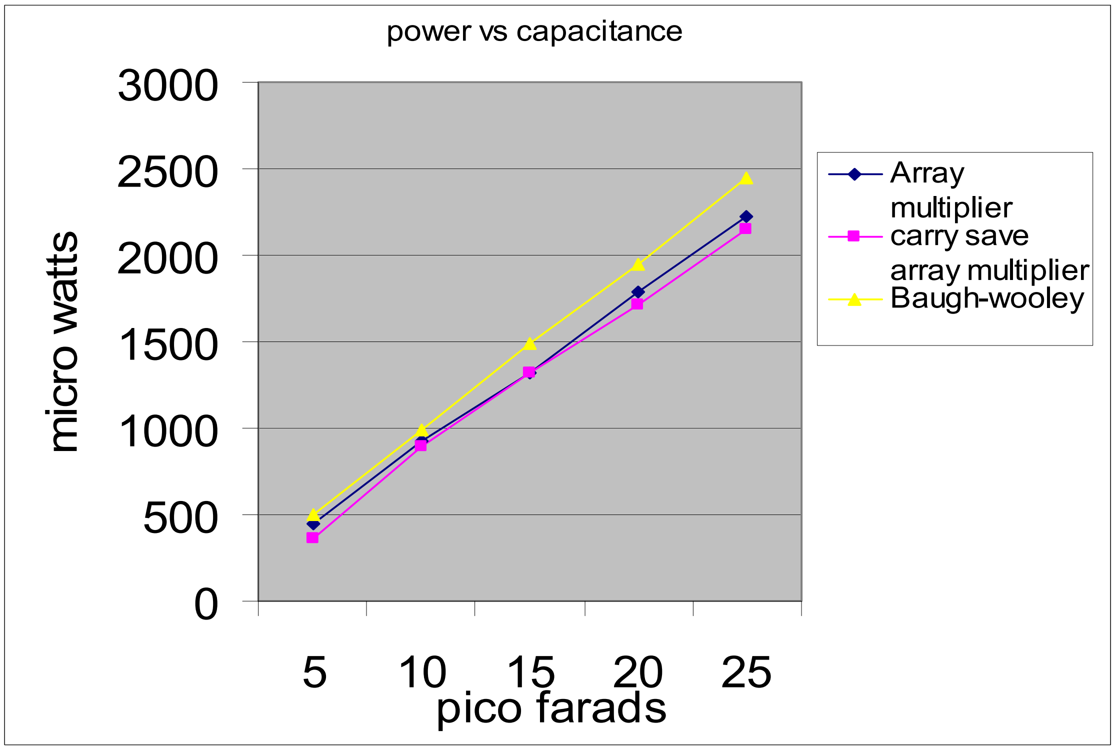


Figure 22: Un-pipelined multipliers comparison

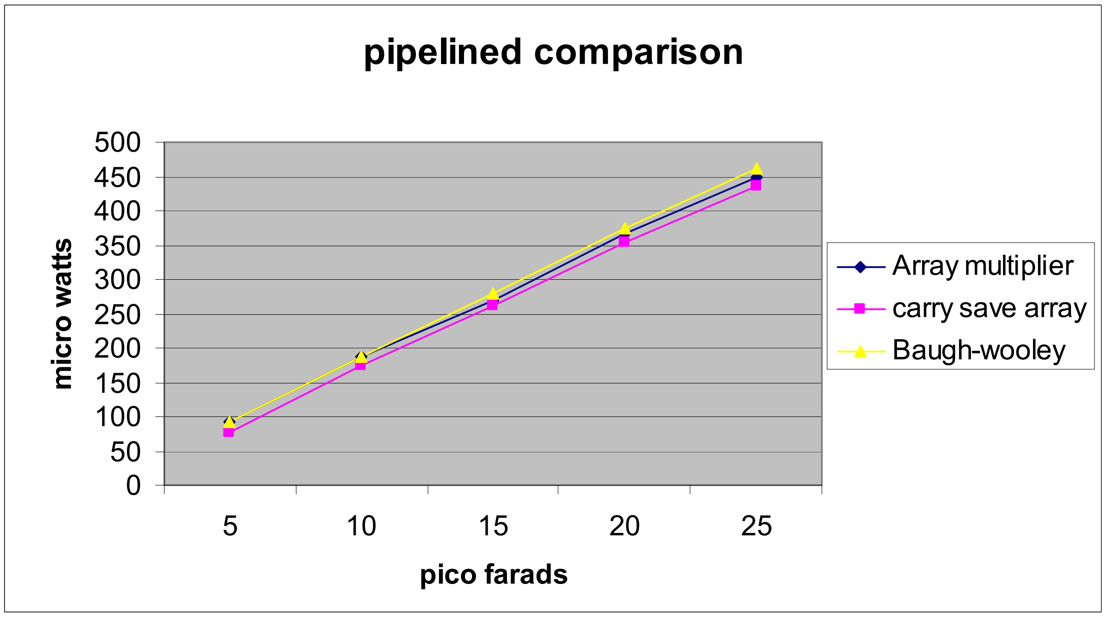


Figure 23: Pipelined multipliers comparison

Table 6: Comparison table

| | | **Un-pipelined Multipliers 4x4 bit** | | | | |
|---|---|---|---|---|---|---|
| | capacitance in pico farads | 5 | 10 | 15 | 20 | 25 |
| | | **Power Dissipation in micro watts** | | | | |
| Array | | 445.5 | 924 | 1320 | 1782 | 2227.5 |
| CSA | | 356.4 | 891 | 1320 | 1716 | 2145 |
| Baugh-Wooley | | 495 | 990 | 1485 | 1947 | 2442 |
| | | | | | | |
| | | | | | | |
| | | **supply voltage 3.3v** | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | **pipelined multiplication** | | | |
| | | | | | | |
| Array | | 92.5 | 187.5 | 270 | 367.5 | 450 |
| CSA | | 77.5 | 175 | 262.5 | 355 | 435 |
| Baugh-wooley | | 92.5 | 187.5 | 280 | 375 | 462.5 |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | **supply voltage 2.5v** | | | | |

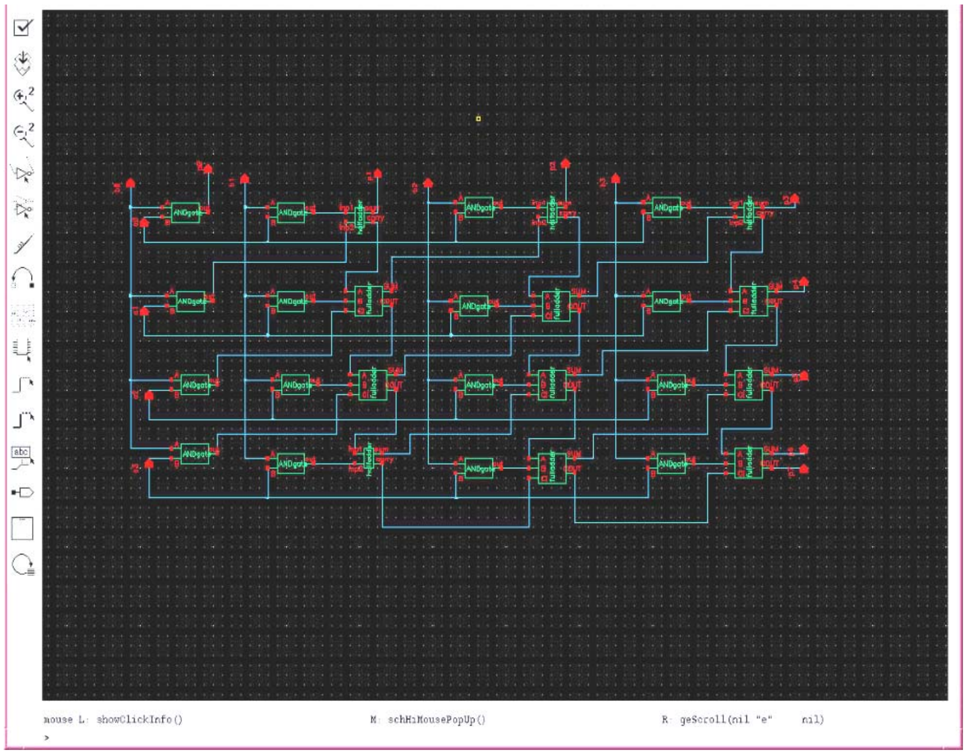

```
mouse L: showClickInfo()          M: schHiMousePopUp()          R: geScroll(nil "e"     nil)
>
```

Figure 24: 4x4 Array Multiplier Schematic

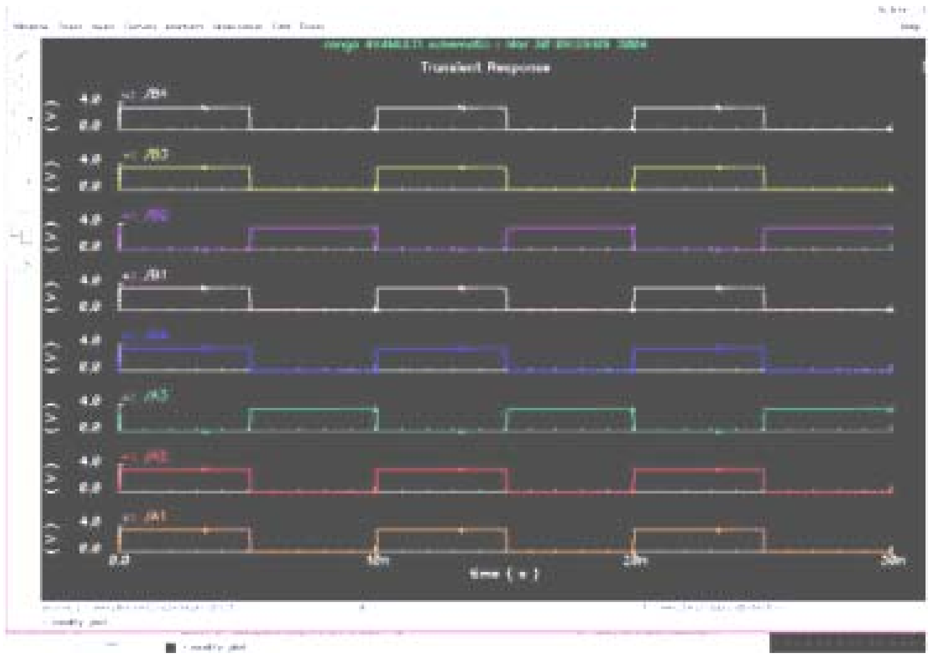


Figure 25: Array Multiplier i/p Waveform

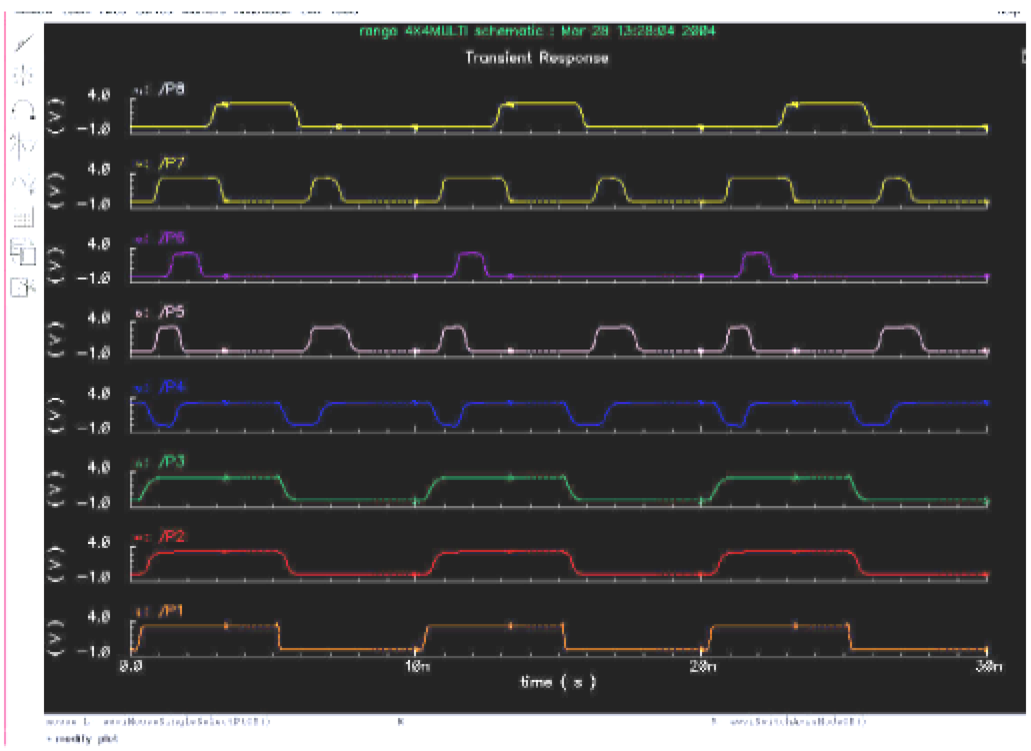


Figure 26: Array Multiplier o/p Waveform

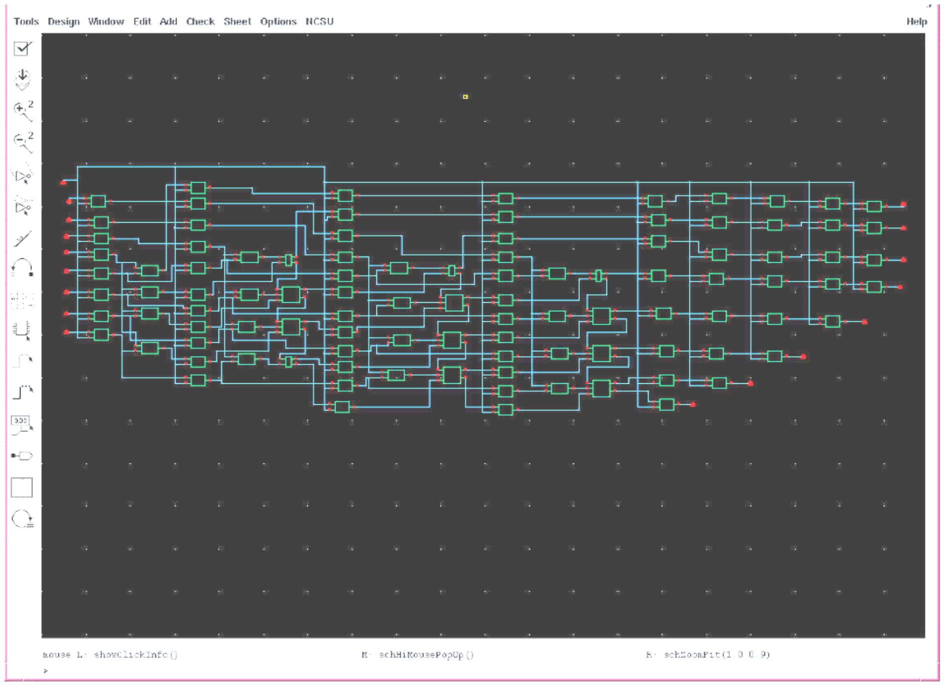


Figure 27: Pipelined 4x4Array Multiplier

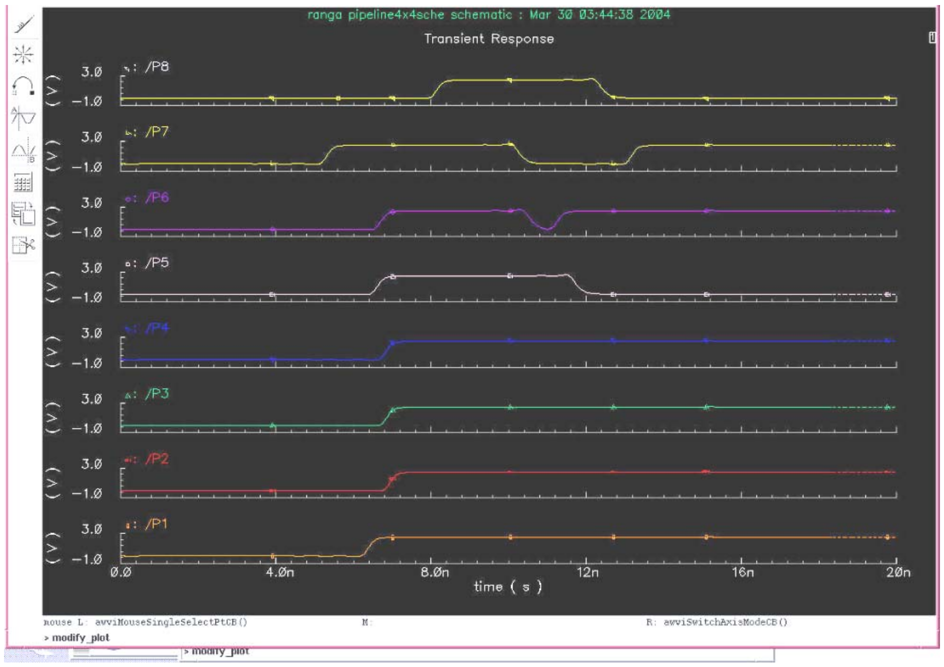


Figure 28: Pipelined Array o/p Waveform

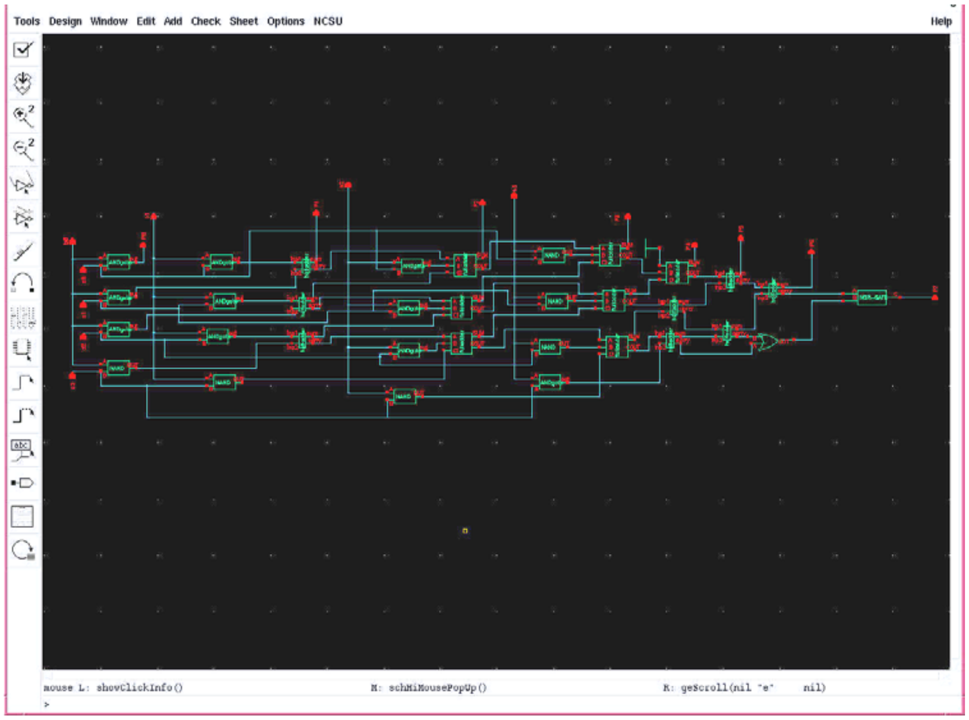


Figure 29: 4x4 Baugh-Wooley Multiplier

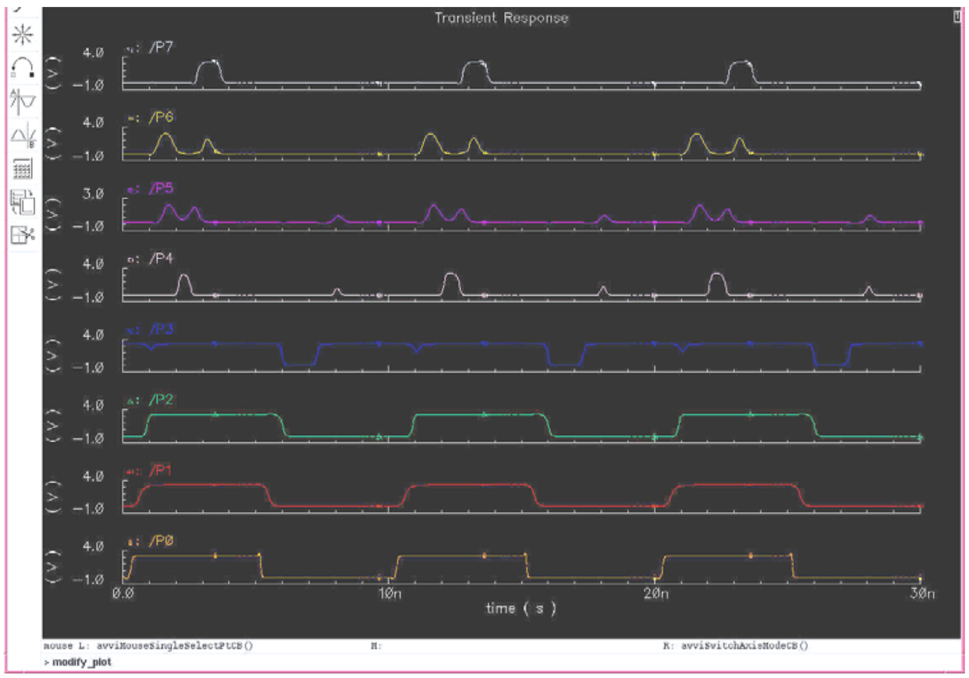


Figure 30: Baugh-Wooley o/p Waveform

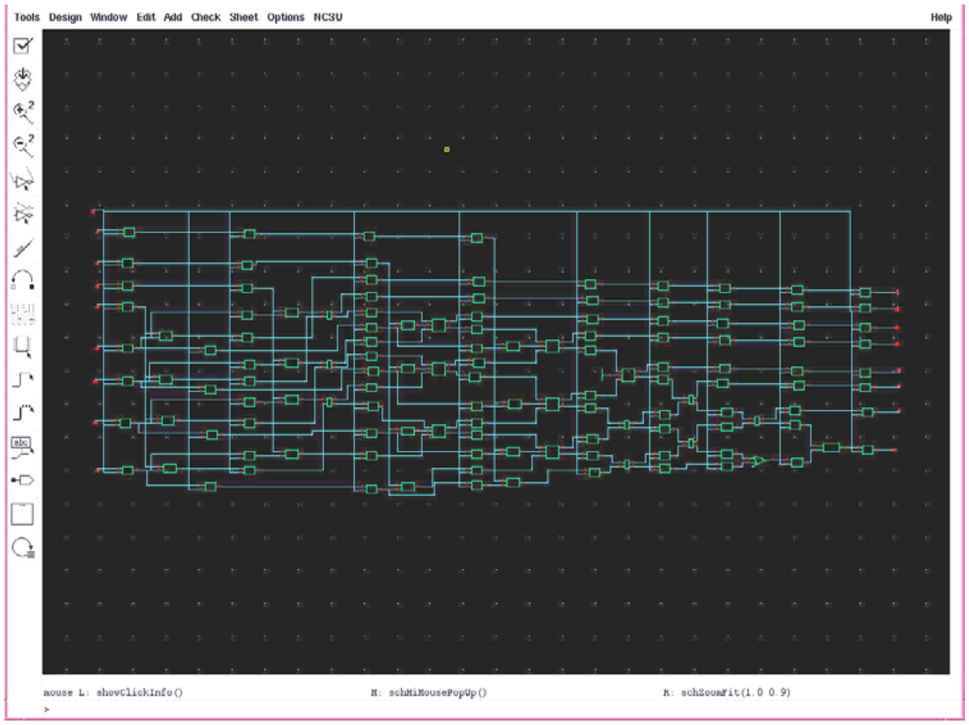


Figure 31: 4x4 Pipelined Baugh-Wooley Schematic

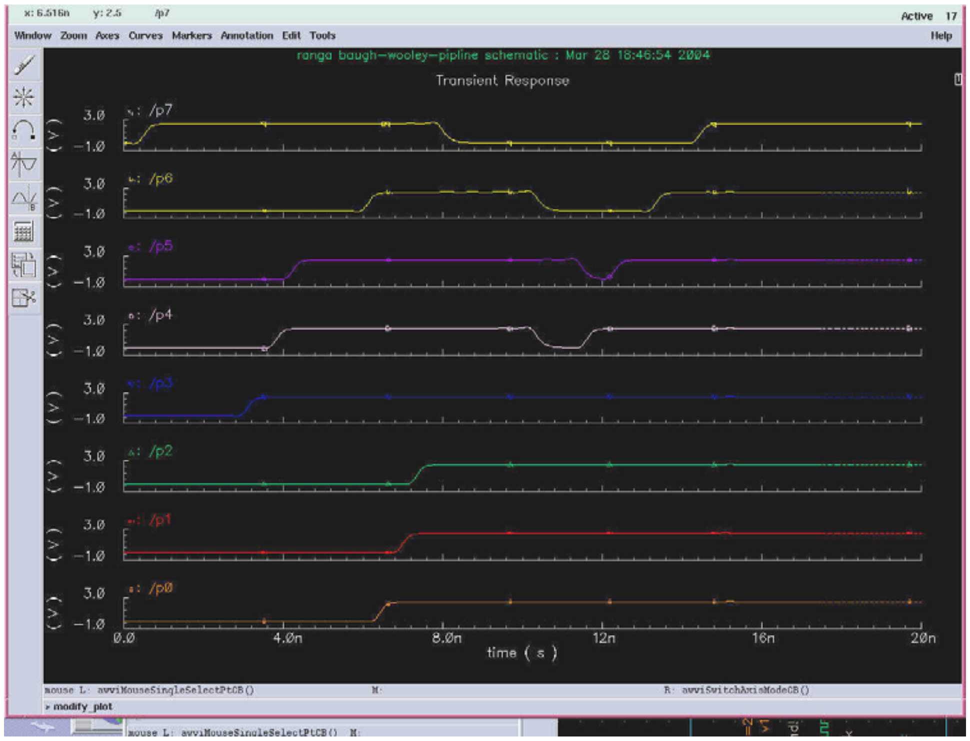


Figure 32: Pipelined Baugh-Wooley o/p Waveform

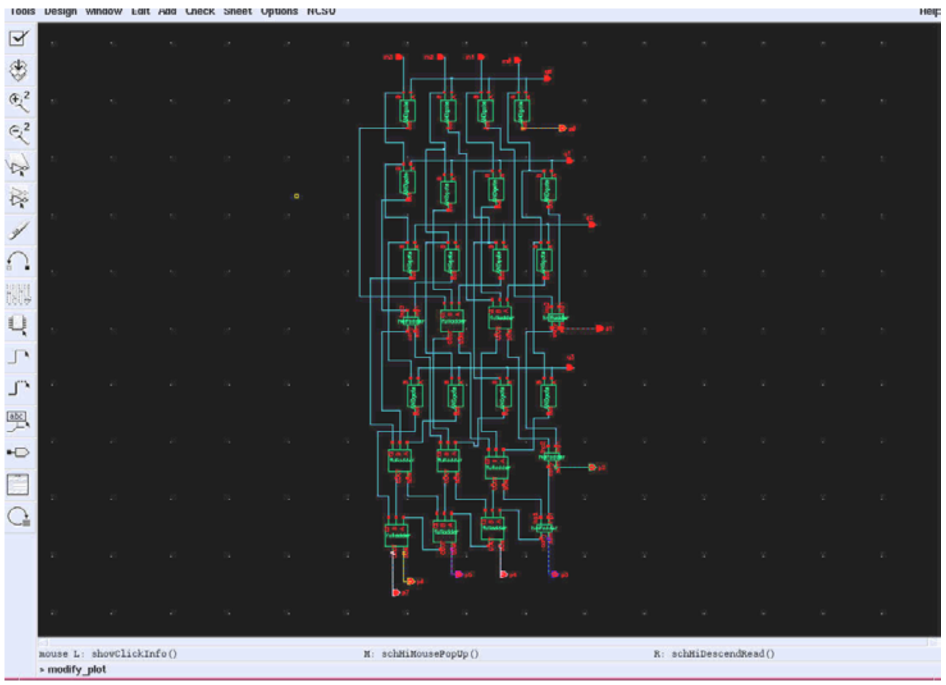


Figure 33: 4x4 Carry Save Adder Schematic

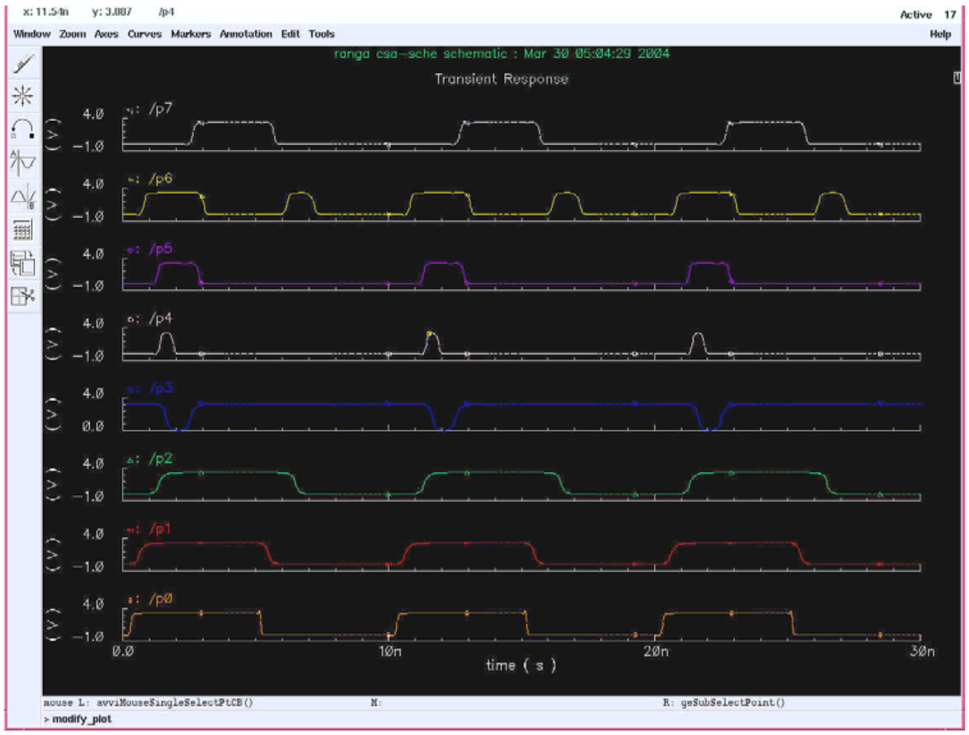


Figure 34: CSA o/p Waveform

# REFERENCES

1) J.S.Yuan et al, “Power-aware pipelined Multiplier Design Based on 2-Dimensional Pipelining Gating.” In proc. of ACM, pp 64-67, 2003

2) L.Benini et al, “A novel approach to cost-effective estimate of power dissipation in CMOS ICs”. In proc. of IEEE, pp 354-360, 1993

3) Manjit Borah et al, “Transistor sizing for minimizing power consumption of CMOS circuits under Delay constraint.” In proc. of IEEE, pp1-6.

4) Dimitrios Soudris et al, “Designing CMOS circuits for Low power”. Kluwer Academic Publishers, 2002.

5) Behrooz parhami, “Computer Arithmetic-Algorithms and Hardware Designs”. Oxford University Press, 2000.

6) Keshab k.Parhi, “VLSI Digital signal processing systems”. Wiley-Interscience Publication, 1999.

7) Harold S.Stone, “High-performance Computer Architecture”. Addison-Wesley Publication, 1993.

8) Tobias G.Noll et al, “A Pipelined 330-Mhz Multiplier”, IEEE journal of solid-state circuits, vol sc-21, 1986.

9) Alexander Taubin et al, “Design of Delay-Insensitive Three Dimensional Pipeline Array Multiplier”, In proc. of IEEE, pp 104-111, 2002.

10) Ayman A.Fayed et al, “A Novel Architecture for Low Power Design of Parallel Multipliers”, In proc. of IEEE, pp149-154, 2001.

11) David Garrett et al, “Challenges in Clock-gating for a Low Power ASIC Methodology”, In Proc. of ACM, pp176-181, 1999.

12) Hai Li et al, "Deterministic Clock-gating for Micro Processor Power Reduction", In proc. of HPCA, pp 113-122, 2003.

13) M.Golden et al, "Comparison of two common pipeline structures", In proc. of IEEE, pp 161-167, 1996.

14) Thomas K.Callaway et al, "Power-Delay Characteristics of CMOS Multipliers", In proc. of IEEE, pp 26-32, 1997.

15) Qing Wu, "Clock-gating and its application to low power design of sequential circuits", In proc. of IEEE, pp 415-420, 2000.

16) Anantha P.Chandrakasan et al, "Minimizing power consumption in digital CMOS Circuits", In proc. of IEEE, pp-498-523, 1995.

17) www.hardwarecentral.com/hardwarecentral/ tutorials/2427/1/

18) N.H.E.Weste et al, "Principles of CMOS VLSI Design-A System Perspective", Addison-Wesley, 2nd edition, 1992.

19) J.M.Rabaey, "Digital Integrated Circuits- A Design Perspective", Prentice Hall, 1996.

20) Jan Rabaey et al, "Pipelining: Just another transformation", In proc. of IEEE, pp163-175,1992.

21) Robert Rogenmoser et al, "A 375 MHZ 1 µm CMOS 8-Bit Mulitplier", Symposium on VLSI circuit digest of technical papers, pp 13-14, 1995.

22) Michael S.Chang et al, "An 86 mw, 80 M sample/sec, 10-bit Pipeline ADC", In proc. of IEEE, 2002.

23) Anders Berkeman et al, "A Low Logic Depth Complex Multiplier Using Distributed Arithmetic", In proc. of IEEE, pp 656-659, 2000.

24) Fabian Klass et al, “A 16x16 bit static CMOS Wave pipelined multiplier”, In proc. of IEEE, pp 1443-146, 1994.

25) Sam S.Appleton, “A new method for asynchronous pipeline control”, In proc. of IEEE, pp 100-104, 1997.